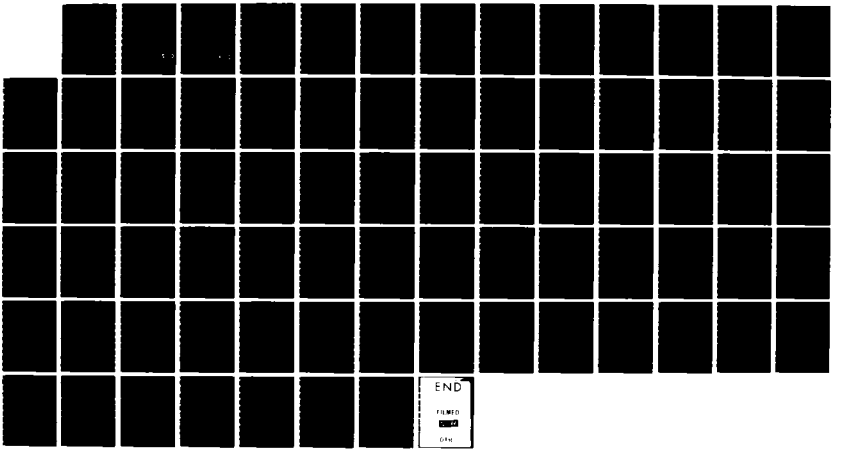
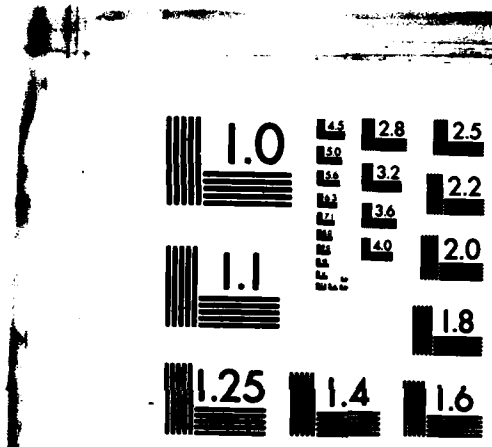
AD-A137 301 PERFORMANCE EVALUATION OF THE INFOPLEX DATABASE COMPUTER USING OPERATIONAL ANALYSIS(U) ALFRED P SLOAN SCHOOL OF MANAGEMENT CAMBRIDGE MA Y Y WANG ET AL. 1/1

UNCLASSIFIED APR 81 TR-13 N00039-78-G-0160 F/G 9/2 NL





(12)

Performance Evaluation of The INFOPLEX Database Computer Using Operational Analysis

Technical Report #13

Y-Y R. Wang

S.E. Madnick

April 1981

Center for Information Systems Research

Massachusetts Institute of Technology
Sloan School of Management
77 Massachusetts Avenue
Cambridge, Massachusetts, 02139

Contract Number N00039-78-G-0160
Internal Report Number M010-8109- 13
Deliverable Number 4

Performance Evaluation of The INFOPLEX Database Computer Using Operational Analysis

Technical Report #13

Y-Y R. Wang

S.E. Madnick

April 1981

Principal Investigator:
Professor Stuart E. Madnick

Prepared for:
Naval Electronics Systems Command
Washington, D.C.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report 13	2. GOVT ACCESSION NO. A137301	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Performance Evaluation of the INFOPLEX Database Computer using Operational Analysis.		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER M010-8109-13
7. AUTHOR(s) Y-Y R. Wang S.E. Madnick		8. CONTRACT OR GRANT NUMBER(s) N00039-78-G-0160
9. PERFORMING ORGANIZATION NAME AND ADDRESS Sloan School of Management M.I.T., Camb., MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE April 1981 13. NUMBER OF PAGES 71
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Performance Evaluation, Simulation, Analytic Modeling, Operational Analysis, Queueing Theory.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The objective of this study is to assess the applicability of analytic techniques to the INFOPLEX database computer. This report builds on earlier work which has used simulation to evaluate the performance of the INFOPLEX data storage hierarchy. Simulation, though accurate, is not cost effective for exploring different design alternatives. Preliminary results from this study indicate that cost effective tools can be developed to —

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

analyze different architecture designs so that optimal design alternatives can be identified early in the design process. The model used in this report produces ceiling throughput to systems under investigation for overhead due to unbalanced flow is ignored. Tighter bounds can be obtained by including the unbalanced flow into analytic models.

Accession For	
NTIS GRA&I	☑
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

EXECUTIVE SUMMARY

The need for efficient storage and processing of very large databases to support decision-making and the advances in computer technology have made research and development of database management systems with specialized architectures a very attractive and important area. The INFOPLEX database computer proposed by Madnick applies the theory of hierarchical decomposition to obtain a specialized architecture for database management systems with substantial improvement in performance over conventional architectures.

A key question in the architectural design of the INFOPLEX data storage hierarchy is how to reduce erroneous design decisions by eliminating potential system bottlenecks and assessing system response time as well as system throughput so that performance requirements can be met. Simulation and analytic modeling are two primary approaches to answer the question. Simulation, although more accurate than the analytic approach, is not cost effective. Preliminary analytic results in this report indicate that analytic modeling techniques are applicable as well as cost effective to the performance evaluation of the INFOPLEX data storage hierarchy. The results obtained from operational analysis and those from RESQ are identical; when comparing with simulation, the results are comparable to within a factor of two. This indicates that analytic techniques are consistent but too gross to incorporate some primary effects. A closer examination reveals that overhead due to unbalanced flow is not included. Future reports will address this issue.

Table of Contents

1 Introduction

1.1 The INFOPLEX Data Base Computer

The need for efficient storage and processing of very large databases to support decision-making coupled with advances in computer hardware and software technology have made research and development of specialized architectures for database management a very attractive and important area.

The INFOPLEX data base computer proposed by Madnick applies the theory of hierarchical decomposition to obtain a specialized architecture for database management with substantial improvement in performance and reliability over conventional architectures. The storage subsystem of INFOPLEX is realized using a data storage hierarchy. A data storage hierarchy is a storage subsystem designed specifically for managing the storage and retrieval of very large databases using storage devices with different cost/performance characteristics arranged in a hierarchy. It makes use of locality of data reference to realize a low cost storage subsystem with very large capacity and small access time. (Lam79)

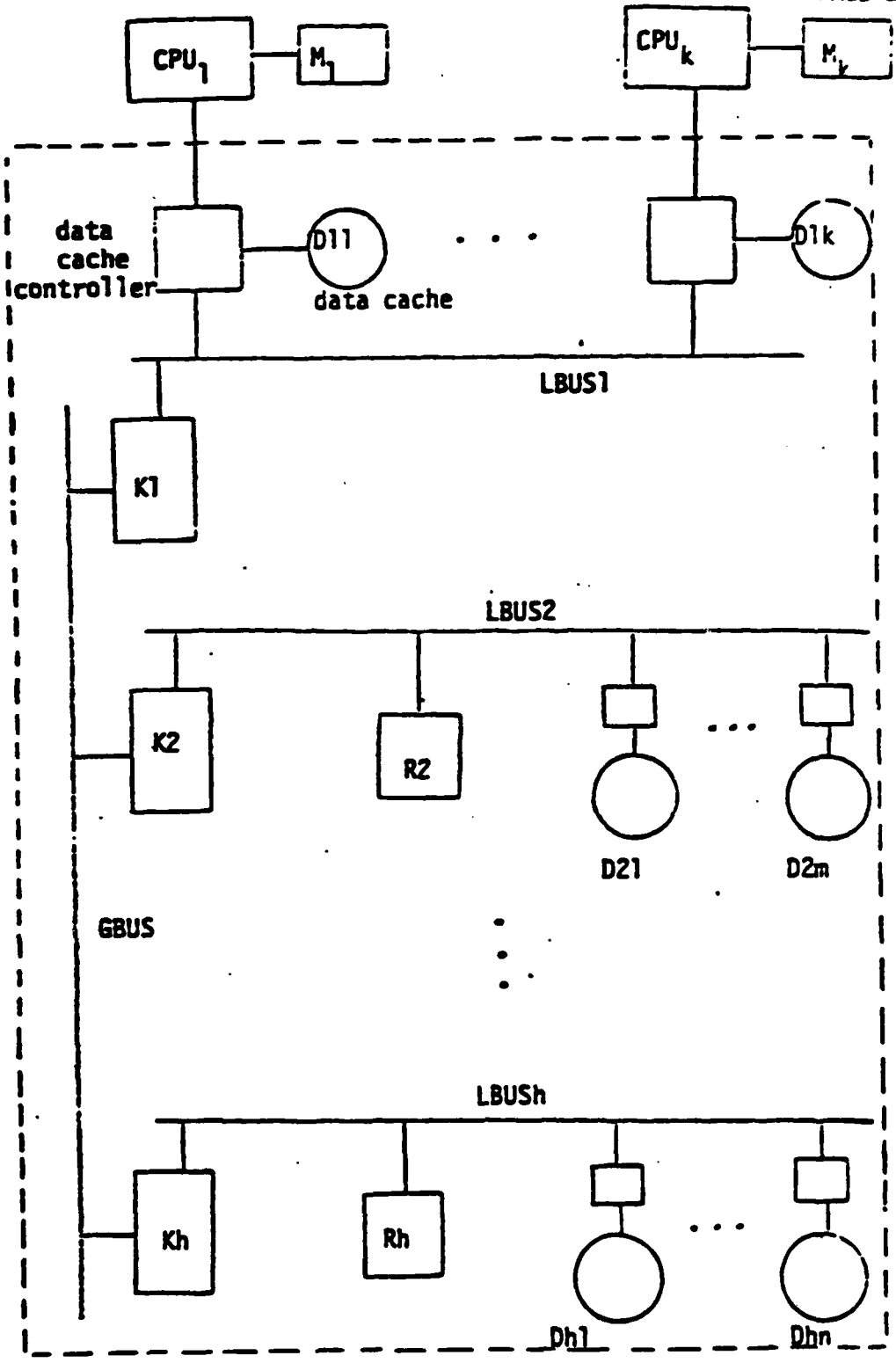

CPU1
M1
CPUk
Mk
data
cache
controller
D11
data cache
D1k
LBUS1
K1
LBUS2
K2
R2
D21
D2m
GBUS
LBUSh
Kh
Rh
Dh1
Dhn

Figure 1.1

1.2 Structure and Algorithms of DSH-11

A design of the general structure of the INFOPLEX Data Storage hierarchy is described in (Lam and Madnick,1979b). A simplified design of the INFOPLEX Data Storage hierarchy, referred to as DSH-11, is described in (Lam and Madnick,1979c). Detailed protocols for supporting the read-through and store-behind operations in DSH-11 are also presented in (Lam and Madnick,1979c). The DSH-11 structure and the DSH-11 algorithms are briefly reviewed here :

The DSH-11 structure is illustrated in Figure 1.1. The highest performance storage level, L(1), consists of the data caches. Each data cache corresponds to a DSH-11 memory port that connects to a processor. All the data caches share a local bus, LBUS1. There is a storage level controller, K1, that serves as the communication gateway between L(1) and lower storage lebvels.

A typical storage level, L(i), for i greater than 1, consists of a storage level controller, Ki, a memory request processor, Ri, and a number of storage device modules, Di1, .., Dim. All these modules share a local bus, LBUSi. Ki is the communication gateway between L(i) and other storage levels. Ri maintains a directory of all the data in L(i). Dij performs the actual storage and retrieval of data.

The global bus, GBUS, connects all the storage level controllers of the storage levels. All communicatio s among storage levels make use of the GBUS.

DSH-11 makes use of the read-through and two-level-store-behind data movement algorithms. A request to read a data item is handled by a data cache, it is retrieved and sent to the processor. If the data item is not in the data cache, the request is passed down to lower storage levels, one by one. At each storage level, the memory request processor searches its directory to determine if the addressed data item is in that level. When the addressed data item is found at a storage level, a block of data containing the addressed data item is broadcasted to all upper storage levels. Each upper storage level then extracts a subblock from the broadcast that contains the addressed data item. The subblock is stored in the storage level. To accomodate an incoming block, an existing block may have to be evicted from a storage level. The way evicted blocks are handled is referred to as the overflow handling strategy.

In a write operation, the data block to be updated is first read into the data cache. After the data block is updated, the data block is sent to the next lower storage level which will update the larger block that contains the data block. Thus, the effect of the update is propagated to lower storage levels. The two-level-store-behind strategy ensures that proper acknowledges are obtained at a given storage level that indicates an updated block has been propagated at least two storage levels down the hierarchy. Thus, at least two copies of the updated data exist at all times.

1.3 Motivation of the Study

A key issue in the INFOPLEX design is performance evaluation. Because of the complexity involved in the design of the INFOPLEX system, it is not intuitively obvious how the performance of a model will be given certain design specifications. Various techniques have been employed to do performance evaluation (Lucas71, also INFOTECH state of the art report, performance modelling & prediction Vol.1). Simulation and analytic models, among others, were found most useful in evaluating the design alternatives of INFOPLEX.

The simulation technique was employed (Lam79) to obtain various performance statistics which provided insights about INFOPLEX. The results were useful, but it was very expensive to explore various design issues using this approach. Buzen has indicated that : (Buzen75)

"... A new family of analytic model has been developed which is based on the theory of queueing networks. These models are sufficiently rich to represent all the essential components of almost any multi-programmed computer system. In many cases, queueing network models have proven to be several orders of magnitude more cost effective than conventional simulation models for a variety of performance evaluation applications ..."

It seems logical to adopt analytic queueing network approach as

a design tool for the performance evaluation of the INFOPLEX data base computer (Madnick80). There are two problem areas in using a queueing network model, namely :

a. How to map system and workload features into a queueing network?

b. How to solve queueing network problems?

The first area is called modeling; the second one deals with suitable methods of solution. The modeling question is least well understood. On the one hand, we find extremely simplified analytic models; on the other hand, we see the designers resorting to costly, highly imitative simulations(Reiser78). The major thrust of this paper is to develop a pragmatic analytic tool for evaluating the performance statistics of the INFOPLEX data base computer. Several objectives were pursued :

1. To be intuitively understandable.

2. To be cost effective.

3. To yield sufficient accuracy for many performance questions.

Buzen's Operational Analysis framework was applied to model the DSH-11 subsystems of the INFOPLEX data base computer.

1.4 OPERational Analysis(OPERA)

The traditional approach to deriving queueing network results depends on assumptions used in the the theory of stochastic processes:

* The system is modeled by a stationary stochastic process.

* Jobs are stochastically independent.

* Transitions from device to device are Markovian.

* The system is in stochastic equilibrium.

* The service time at each device has an exponential distribution; and

* The system is ergodic, i.e. long-term averages converge to the values computed for stochastic equilibrium.

The theory of queueing network based on these assumptions is usually called "markovian queueing network theory" (Klein75). However, some of these concepts such as "equilibrium" or "stationarity", can not be proved to hold by observing the system in a finite time period. In fact, most can be disproved empirically - for example, in a computer system, parameters change over time, jobs are dependent, device to device transactions do not follow Markov chains, and service distributions are seldom exponential(Buzen78c).

By using a different set of assumptions, operational equations can be derived and they are likely to hold in actual systems. This has been proved to be true(Buzen76a,b,c,78c, and Denn77), and is referred to as operational analysis. Chapter 3 will review OPERA and use it to model INFOPLEX systems.

1.5 Multi-Transaction-Type, Multi-Class Modeling Technique

As mentioned in section 1.3, there are two problem areas to be considered in using a queueing network model. OPERA provides us with a framework to solving queueing network problems. But the problem of mapping system and workload features into a queueing network remains unsolved in the INFOPLEX environment because of the complexity involved in the multi-level structure of DSH-11. Different modeling techniques can be employed to exploit the OPERA's power. The concept (Baskett75) of multi-class customers, among others, is found to be useful in modeling the DSH-11 models. Since multi-transaction types (read and write transactions for example) are present in the INFOPLEX environment, the techniques to model an multi-transaction-type system are also discussed. The multi-class concept allows a device to have several different classes of transactions. For each class, there are distinct values for the service times and the routing frequencies. Each job belongs to exactly one class, and each class is local to a specific device. Each class is independent of one another in the sense that they do not "talk" to one another.

The multi-transaction-type concept coupled with the multi-class concept enables a modeler to decompose different types of transactions into subsystems. After developing a queueing network model for each type of transaction, the different submodels can then be integrated to form the desired queueing network model. 3.2.2 describes the multi-class model.

It is interesting that by using the multi-class technique for

modeling INFOPLEX, one can get a diagonally structured transition matrix for the system, therefore can compute visit ratios(the number of times a job visits a device) by inspection. Furthermore, it is interesting to note that the ceiling throughput can be estimated by a simple formula derived from the reasoning of the multi-class, multi-transaction-type concept.

1.6 Structure of the Paper

Chapter 2 surveys state-of-the-art analytic models. Chapter 3 reports modifications made to Buzen's work(Buzen78c), key differences, computational procedures, and the results obtained. Chapter 4 reports RESQ's(Research Queueing Analyzer Program Package) result which confirms the result of the OPERational Approach (OPERA). Chapter 5 compares OPERA's results with Lam's simulation results using comparable P1L3(1 processor, 3 levels) model of INFOPLEX with comparable inputs. Chapter 6 extends the OPERA's results to a general DSH-11 model with R proportion of READ transactions, L levels, P loaclity reference, and bottleneck service time S(b). Chapter 7 concludes this report and indicates future directions along this line of work.

1.7 Summary of the Paper

The inventions and innovations of computer hardware and software technologies together with the needs for high performance, high reliability, and nearly infinite storage capacity computers stimulated research and development of specialized architectures for

database management systems.

The INFOPLEX data base computer proposed by Madnick applies the theory of hierarchical decomposition to obtain a specialized architecture for database management with substantial improvement in performance and reliability over conventional architectures.

A key issue in the INFOPLEX design is performance evaluation. Because of the complexity involved in the design of the INFOPLEX system, it is not intuitively obvious how the performance of a model will be affected by different design choices.

This paper provides a cost effective and intuitively appealing solution technique for evaluating the performance of the complex INFOPLEX data base computer. By adopting the operational approach, an intuitive argument and a cost effective solution can be obtained. By using the multi-class, multi-transaction-type modeling technique, a diagonally structured transition matrix for the INFOPLEX model can be constructed which leads to the inspectability of the visit ratios to all devices. From these visit ratios, one can compute the bottleneck device of a closed system which provides information about the system's throughput. The system's response time can be obtained from Little's formula once the system's throughput and number of customers in the system are known. The major achievements of this paper can be summarized as follows :

- provide an intuitive and analytically tractable method to evaluate the performance of the complex INFOPLEX data base

computer(chapter 3).

- obtain a diagonally structured transition matrix which simplifies the computation of visit ratios for INFOPLEX by using the multi-transaction-type, multi-class modeling technique(chapter 3).

- validate OPERA results by the traditional stochastic approach implemented by RESQ and by the GPSS simulation results obtained by (Lam79)(chapter 4 and 5).

- provide a quick and handy way to compute the maximum throughput of any INFOPLEX model when given a set of (locality reference, number of levels, proportion of READ transactions, slowest device service time)(chapter 6).

- provide a framework for future performance analysis of INFOPLEX design alternatives. More complicated model can be obtained by refining current model to accomodate overflow-handling, broadcasting, priorities, etc...

2 Analytic Model, State of the Art

The theory of queueing networks has been developed for more than two decades. Queueing network models are becoming popular as models for performance analysis of computer systems and communication networks. (Buzen78c) did an excellent summary of the evolvement:

"... In 1957, Jackson published an analysis of a multiple device system wherein each device contained one or more parallel servers and jobs could enter or exit the system anywhere. In 1963 Jackson extended his analysis to open and closed systems with local load-dependent service rates at all devices. In 1967, Gordon and Newell simplified the notational structure of these results for the special case of closed system. Baskett,et al, extended the results to include different queueing disciplines, multiple classes of jobs, and nonexponential service distribution(Baskett75). The first successful application of a network model to a computer system came in 1965 when Scherr used the classical machine repairman model to analyze the MIT time sharing system,CTSS(Sche67). However, the Jackson-Gordon-Newell theory lay dormant until 1971 when Buzen introduced the central server model and fast computational algorithms for these models. Working independently, Moore showed that queueing network models could predict the response times on the Michgan Terminal System to within 10 percent. Extensive validations since 1971 have verified that these models reproduce observed performance quantities with remarkable accuracy..."

(Buzen78a) proposed OPERA as an alternative to stochastic modelling. A framework of the OPERA models is detailed in (Buzen78b). In that paper, the operational approach to queueing network model is outlined. The operational analysis assumptions can be tested, the analysis is much more intuitive and tractable by human mind, and there are good reasons to believe that they often hold.

The applications of analytic models to the performance analysis of computer systems has become more and more prevalent. Some researchers continue to use the traditional approach(Lavenberg80, Reiser80, Chandy80, and Bard80 for instances); others try to use operational analysis to study the properties of queueing netwrok models (Buzen,

Denning80). Since the introduction of operational analysis in 1976 as an alternative to stochastic modeling, many informal, intuitive arguments used to motivate stochastic theorems become rigorous proofs in the formal context of operational analysis. Besides simplifying derivations, operational analysis extends stochastic theorems by demonstrating their validity in cases where conventional stochastic assumptions cannot be justified. Moreover, operational analysis has led to new results about sensitivity factors and error bounds. These results are particularly valuable for prediction because the future validity of operational(or stochastic) assumptions is usually uncertain(Buzen80).

From the application point of view, OPERA has a lot of significant advantages over the traditional stochastic approach - not only because it gives operationally testable variables, but also for its intuitive arguments which make the results much more convincing to the modeler. With this approach, it is possible to use the queueing network technology with much more confidence and understnading. This paper adopts this approach. Chapter three reviews OPERA and some modelling techniques for DSH-11.

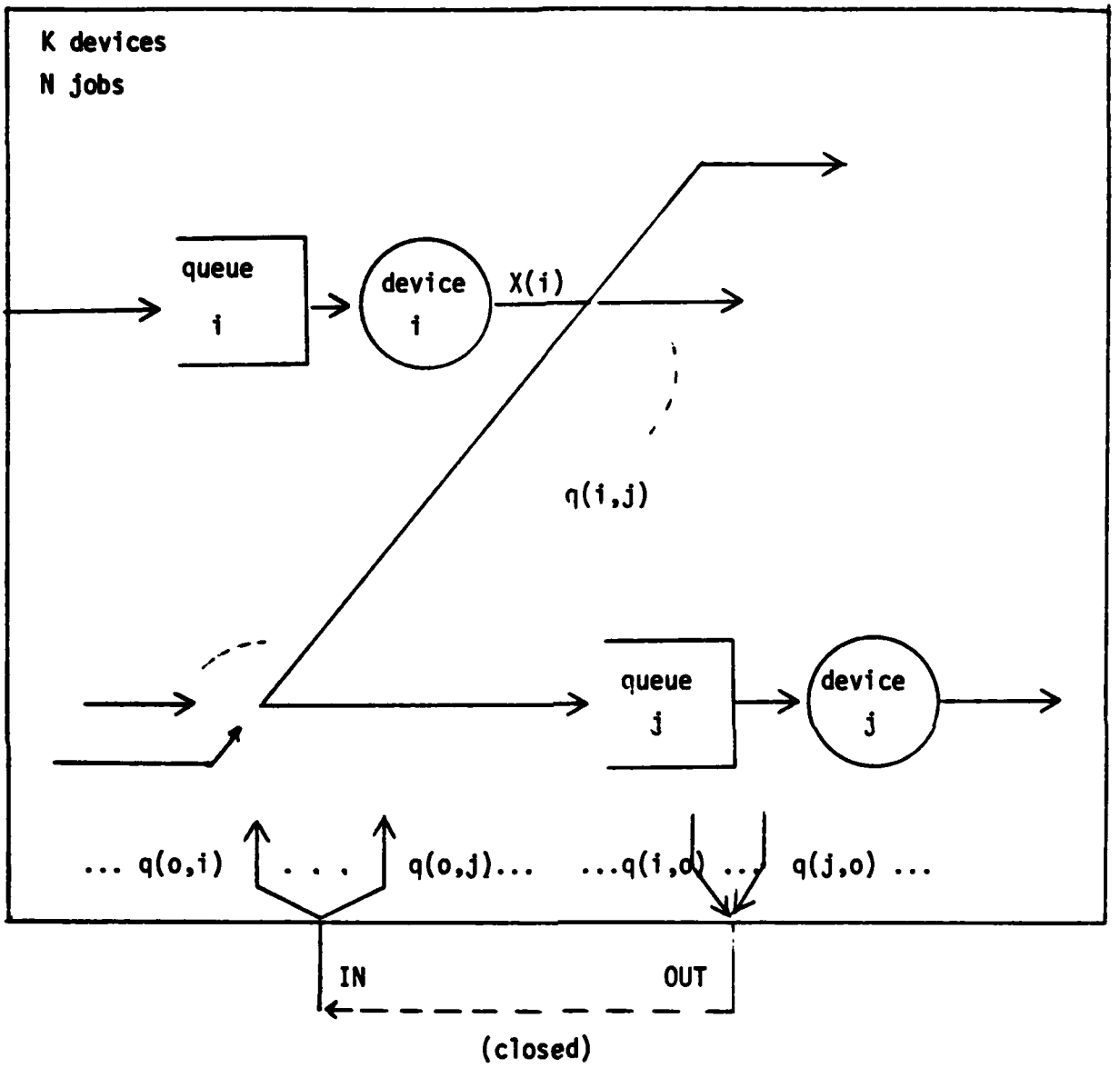
3 OPERational Analysis Approach (OPERA)

In this chapter, we first discuss the framework proposed by Buzen, and the computational procedures for open and closed systems under the INFOPLEX environment. Then the notion of mulit-class queueing network model for operational analysis is introduced. The notion of multi-class queueing network model uses the same idea as single-class queueing network models but has several classes of transactions for each device. Algorithms are developed to fit the multi-class notion. The multi-class concept leads itself to a diagonally structured transition matrix, henceforth simplifies the computational complexity because of the inspectability of the structure. Finally we extend the multi-class queueing network model from a single-transaction-type environment to a multi-transaction-type environment and compute the performance statistics of P1L3 model of INFOPLEX.

3.1 Single-Transaction-Type, Single-Class Queueing Network Model.

(Buzen78c) has proposed the framework of the operational analysis of queueing network models. The framework is sufficiently rich to represent all the essential components of almost any multi-programmed computer systems. We use this framework as the basis for the analysis of this paper.

Since it is much simpler to discuss an environment which possesses only a single type of transactions, we introduce the concept of a single-transaction-type model versus a multi-transaction-type model. A multi-transaction-type model is a



Two Devices in a Queueing Network --- Figure 3.1.1

variation of the single-transaction-type model in the sense that it uses the result established by the single-transaction-type model to model the multi-transaction-type environment. For instance, in a central server model, different transaction types can be combined into a single type transaction by using weighted averages as parameters. Since the routes that different types of transactions travel within the central server model are similar, this kind of combination to a single-transaction-type model suffices.

3.1.1 Operational Measures Proposed by Buzen.

3.1.1.1 Model Description

Fig-3.1.1 shows two of K devices in a multi-resource network. A job enters the system at IN. It circulates around in the network, waiting in queues and having services requests processed at various devices. When done, it exits at OUT. The network is operationally connected in that each device is visited at least once by some job during the observation period.

A job is "in queue" at device i if it is waiting for or receiving service there. We let n(i) denote the number of jobs in queue at device i, and N = n(1)+...+n(k) denote the total number of jobs in the system. The system output rate,X(o), is the number of jobs per second leaving the system. If the system is open, X(o) is externally specified and N varies as jobs enter or leave the system. If the

system is closed, the number of jobs N is fixed. This is modeled by connecting the output back to the input, as suggested by the dashed arrow in Fig-3.1.1.

An analysis of an open system assumes that X(o) is known and seeks to characterize the distribution of N. An analysis of a closed system begins with N given and seeks to determine the resulting X(o) along the OUT/IN path. System response time,utilization rate, mean queue length, and response time for each device, are sought in both cases.

3.1.1.2 Basic Notation

K : # of devices in the system, k=1,2,...,K

N : denote the total # of jobs in the system where N = n(1)+n(2)+...+n(k)

X(o): The system output rate. It is the number of jobs per second leaving the system.

T : An observation period that the system is measured.

R(or R(o)) : denotes the system's response time.

For each device k = 1,2,...,K :

n(k) : the number of jobs in queue at device k. It is the queue length at device k; it includes jobs waiting and receiving services

A(k) -- # of arrivals during the observation period.

B(k) -- total busy time (time during which n(k) > 0)

X(k) : Throughput of device k.

R(k) : Response time of device k.

C(i,j) -- # of times a job requests service at device j immediately after completing a service request at device i.

q(i,j) -- routing frequency, the fraction of jobs proceeding next to device j on completing a service request at device i.

If we treat the "outside world" as device "o", we can define also :

A(o,j) -- # of jobs whose first service request is for device j.

C(i,o) -- # of jobs whose last service request is for device i.

a(k) -- arrival rate at device k. /* a(k)=A(k)/T */

a(o,j) -- arrival rate at device j from the "outside world" /* a(o,j)=A(o,j)/T */

Q(k) -- queueing time at device k.

Nbar, nbar(k) ..., denote the mean values of N , n(k) ...

C(k) = C(i,1)+C(i,2)+...+C(i,K)

C(o) = C(0,o)+C(1,o)+...+C(K,o)

S(k) : mean service time of device k

V(k) = X(k)/X(o); the visit ratio of a job to device k.

3.1.1.3 Assumption

The model assumes that no jobs overlap its use of different devices. In practice, few application programs ever achieve more than a few percent overlap between CPU and I/O devices; the error introduced by this assumption is usually not significant. The model also assumes that a device is busy if a request is pending there -- no part of the system can block progress in another part. This

assumption is not met by all real systems; for example, the CPU might be unable to continue if an I/O buffer is full. We will assume that C(o,o)=0 because otherwise there would be jobs that used no resources before departing. However, it is possible that C(k,k) > 0 for any device k since a job could request another burst of service from a device which had just completed a request from that job.

The system must be flow balanced -- i.e. the number of arrivals at a given device must be(almost) the same as the number of departures from that device during the observation period.

The device must be homogeneous -- i.e. the routing of jobs must be independent of local queue lengths, and the mean service time at a given device must not depend on the queue lengths of other devices.

3.1.1.4 Performance Statistics Computation

The number of completions at device k is :

$$C(k) = C(k,0)+C(k,1)+C(k,2)+\ldots+C(k,K) \quad k=1,2,\ldots,K.$$

The number of arrivals to, and departures from the system is:

$$A(o) = A(o,1)+\ldots+A(o,K)$$

$$C(o) = C(1,o)+\ldots+C(K,o)$$

From Fig3.1.1 it is clear that A(o) = C(o) in a closed system. In an open system which reaches steady state but not fully utilized, we also have A(o)=C(o).

In terms of B(k),C(k), four derived operational quantities are defined :

U(k) =B(k)/T.

S(k) = B(k)/C(k)

X(k)= output rate of requests from device k = C(k)/T

Note that, for any k, q(k,o)+q(k,1)+...+q(k,K)=1.

Job Flow Analysis

Given the mean service times S(k) and the routing frequencies q(k,j), how much can we determine about overall device completion rates X(k) or response times R(k) ? These questions are usually approached through the operational hypothesis known as the Principle of Job Flow Balance: For each device k, X(k) is the same as the total input rate to device k. This principle will give a good approximation for observation periods long enough that the difference between arrivals and completions,A(k)-C(k) is small compared to C(k). It will be exact if the initial queue length is the same as the final length. Choosing an observation period so that the initial and final states of every queue are the same is not inappropriate.

When a job flow is balanced, we refer to the X(k) as device throughput. The Flow Balance Principle can be expressed as: C(k)=A(k) k=0,1,...,K.

We can also obtain X(k) = X(o)q(o,k) + X(1)q(1,k) + ...

$+X(K)q(K,k)$ $k=0,1,\ldots,K$

Define $V(k)=X(k)/X(o)$ and call V(k) the visit ratio to device k, we immediately get $V(k) = q(o,k) + V(1)q(1,k) + \ldots + V(k)q(K,k)$ for $k=0,1,\ldots,K$

If the network is open, the value of X(o) is externally specified and these equations will have a unique solution for the unknowns X(k). However, if the network is closed, X(o) is initially unknown, and the equations have no unique solution because the sum of the X(k) equations for k=1,...,K reduces to the X(o) equation. Therefore, in a closed network, there are K independent equations but K+1 unknowns. Nonetheless, the job flow balance equations contain information of considerable value. In particular, when a bottleneck situation occurs, the X(o) of a closed system can be obtained by setting the utilization of the bottleneck device to one.

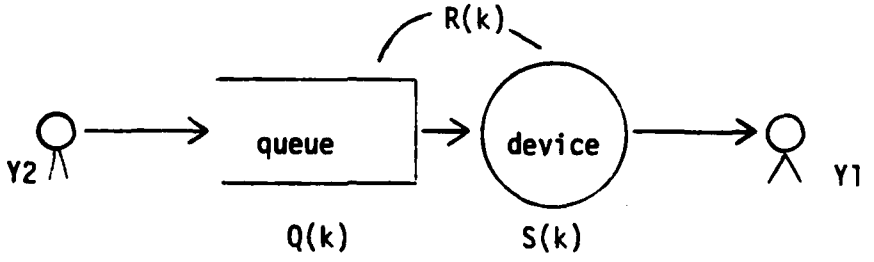
Utilization Rate

$U(k)=B(k)/T=\{C(k)/T\}*\{B(k)/C(k)\}$ holds for each devicee. Therefore,

$$U(k) = X(k) * S(k)$$

Response Time at Each Device

To compute R(k) for device k, consider



$$R(k) = Q(k) + S(k)$$

What is Q(k) ?

Consider the instant a job Y1 departs from device k, and job Y2 arrives at device k :

R(k) = time spent in device k. a(k)*R(k) = number of jobs that arrived after job X arrived and up to the instant job X departs.

{a(k)*R(k)}*S(k) = time to process backlog before job Y2 can be processed. = queueing time for job Y2.

So on average, Q(k) = a(k)*R(k)*S(k)

Since R(k)=Q(k)+S(k), therefore, R(k)=S(k)/{1-a(k)S(k)}. Applying the principle of job flow balance, we have a(k)=X(k). Hence, R(k)=S(k)/{1-X(k)*S(k)} for device k,k=1,2,...,K.

System Response Time

To compute system's response time R, we apply Little's formula to the system and each device :

$$Nbar = R*X(o) \quad , \quad nbar(k) = R(k)*X(k)$$

$$Nbar = nbar(1) + nbar(2) + \ldots + nbar(K) = R(1)*X(1) + R(2)*X(2) + \ldots + R(K)*X(k)$$

$$\text{therefore, } R = Nbar/X(o) = \{R(1)*X(1) + R(2)*X(2) + R(3)*X(3) + \ldots + R(k)*X(k)\}/X(o)$$

Notice that Little's formula does not depend upon any specific assumptions regarding the arrival distribution or the service time distribution; nor does it depend upon the number of servers in the system or upon the particular queueing discipline within the system. In the derivation of Little's formula, the boundary around a queueing system is not precisely defined, therefore the formula can be applied to the entire queueing system as well as to a single queue(Klein75).

3.1.2 Computational Procedure for Closed System

INPUT	STEP	OPERATING EQUATION	OUTPUT
N	1.	$V(o) = 1$	
q(k,j)		compute $V(k)=V(0)q(0,k)+\ldots+V(K)q(K,k)$ $k = 1,2,\ldots,K$	V(k)'s k=1,...,K
S(k)'s	2.	compute $V(b)*S(b) = \max\{(V(k)S(k)\}$	
	3.	$X(o)=1/\{V(b)*S(b)\}$	X(o)
	4.	Compute $U(k)=X(o)V(k)S(k)$, $k=1,\ldots,K$	U(k)'s
	5.	$R= N/X(o)$	R

X(o) is the maximum throughput given N.

R is the corresponding system's response time.

R and X(o) give the performance of the system given the input workload.

U(k) gives us the utilization rate of device k.

3.1.3 Computational Procedure for Open System.

For an open system with single transaction type in steady state which satisfies job flow balance hypothesis, we have the following interesting and intuitive results :

$X(o)=C(o)/T=\{C(o,0)+C(o,1)+\ldots+C(o,K)\}/T=\{A(o,0)+\ldots+A(o,K)\}/T=a(o,0)+\ldots+a(o,K)$

Hence X(o) is externally specified in an open system.

The results are as follows:

INPUT	STEP	OPERATING Equation	OUTPUT
a(o,k)'s	1.	$X(o) = a(o,0)+\ldots+a(o,K)$	X(o) (throughput)
q(k,k)'s	2.	$X(k)=X(o)q(0,k)+X(1)q(1,k)+\ldots+X(K)$ $k=0,\ldots,K$	X(k)'s k=1,...,K

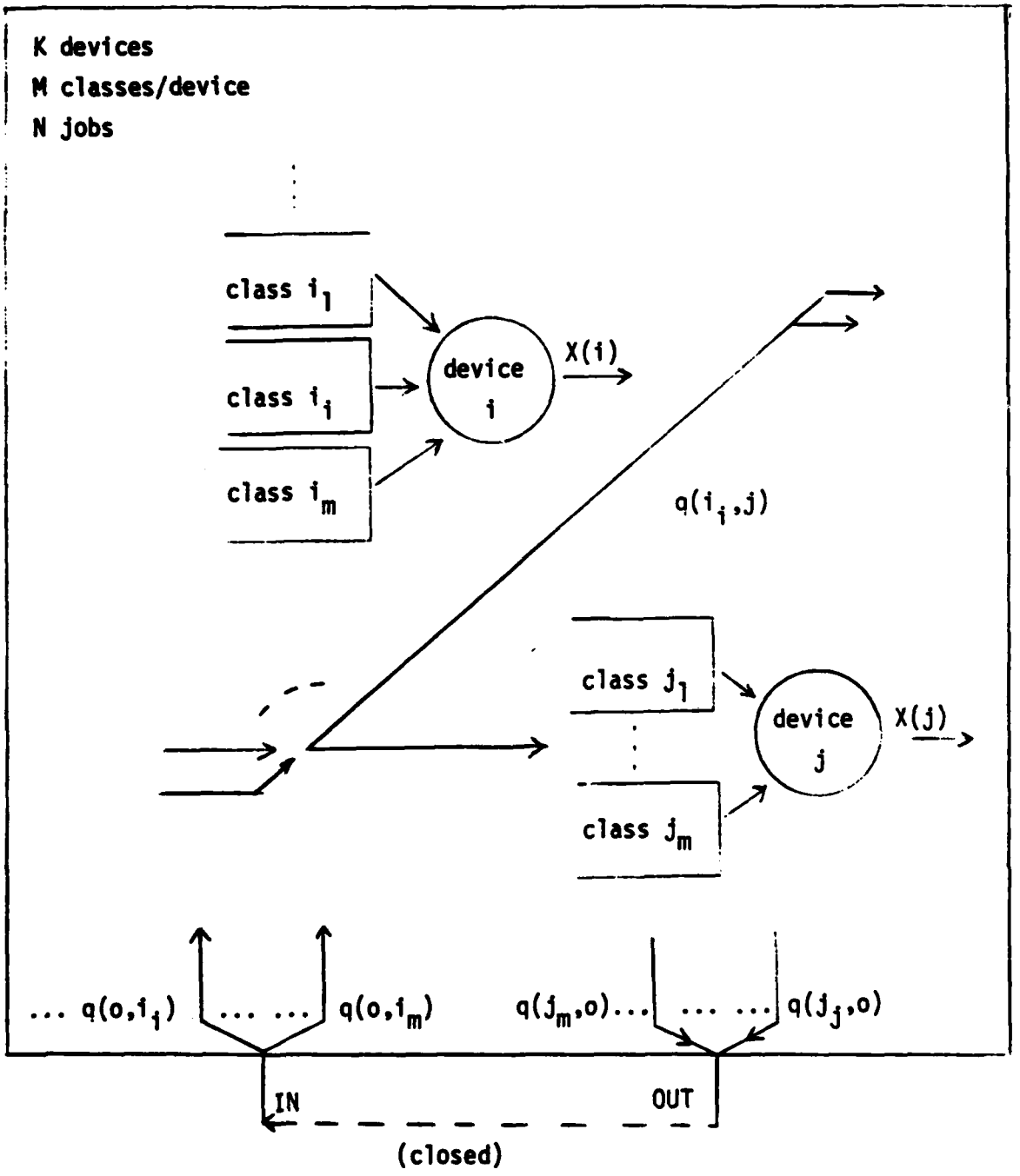


Figure 3.2.1 Two Devices in a Multi-Class Queueing Network

S(k)'s	3.	U(k)=X(k)*S(k) k=1,...,K If U(k) > 1 for some k, then quit.	U(k) > 1.
	4.	R(k)=S(k)/{1-X(k)S(k)} k=1,...,K	R(k) k=1,...,K
	5.	R={R(1)*X(1)+...+R(K)*X(K)}/X(o)	R
	6.	nbar(k)=R(k)X(k) k=1,...,K	nbar(k) k=1,...,K
	7.	Nbar = nbar(1)+...+nbar(K)	Nbar

R is used to measure how fast the system's response is. Note that Nbar=X(o)*R should hold which serves as a check.

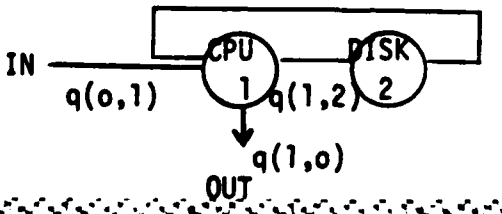
U(k),R(k), and n(k) are used to see how busy a device is. If U(k) and n(k) are large, then chances are we have to improve the performance of that device.

3.2 Single-Transaction-Type, Multi-class Queueing Network Model.

3.2.1 Introduction

The open/closed system computational procedures developed in the previous sections share the same problems that the estimations of S(k) and q(k,j) do not follow the real situation. Re-estimation is necessary in both open and closed systems which decreases an evaluator's confidence in his results.

Consider the CPU-disk example below :



A job visits CPU 4 times and disk 3 times before it exits the system. Since V(o)=1, therefore, V(1)=4,V(2)=3. Hence q(o,1)=1,q(1,o)=.25, q(1,2)=.75 have to be estimated accurately in order to obtain V(1)=4 and V(2)=3. In this case, estimation of q(i,j) is easily done. However , in a general queueing network, the task of estimating q(k,j) is no longer intuitively obvious. Further, an evaluator would also feel very uncomfortable with estimating S(k). In this section, a model with single transaction type using multi-class concept is developed to accommodate this shortcoming.

3.2.2 Model Description

Fig-3.2.1 shows two of the K devices in a multi-device, multi-class per device queueing network. A job enters the system at IN. it circulates around in the network, waiting in queue and having service requests processed at various devices. When done, it exits at OUT. The difference is that each device has m different classes of customers. Each job belongs exactly to one class of the device when it visits the device. Each class has its own service time and routing probabilities. Jobs arrive at a device as member of a particular class. During a visit at a device, a job does not change its class membership. Each class can be given a unique number within the system.

Let's consider the CPU-Disk example again. By applying the multi-class idea, we can model it as in Figure 3.2.2:

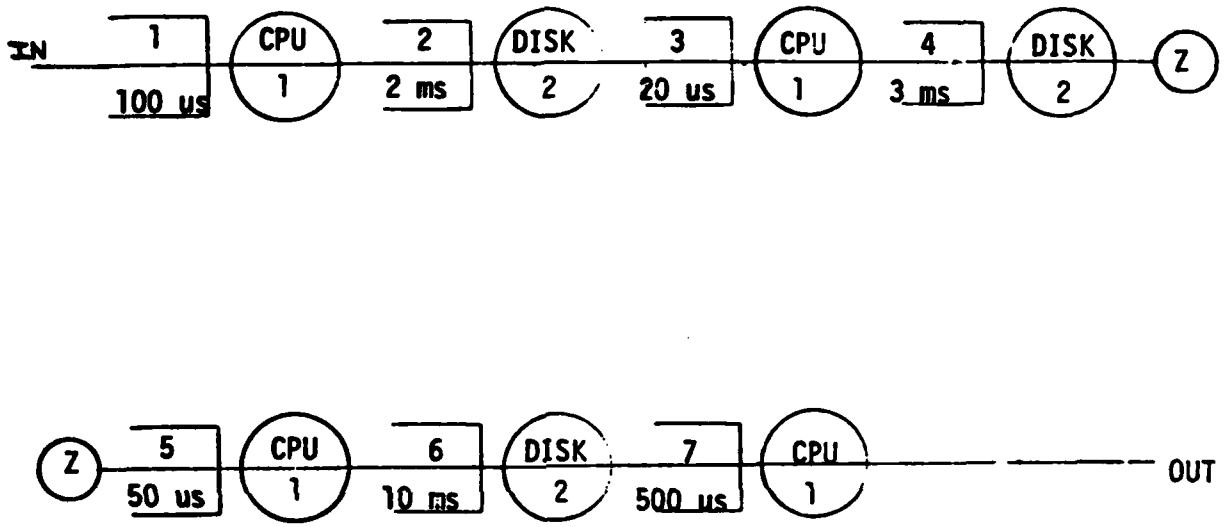

IN
1
100 us
CPU
1
2
2 ms
DISK
2
3
20 us
CPU
1
4
3 ms
DISK
2
Z
Z
5
50 us
CPU
1
6
10 ms
DISK
2
7
500 us
CPU
1
OUT

Figure 3.2.2

It seems that we have complicated the situation, but actually not. This method offers us the following nice and neat properties:

1. It is more natural: a transaction will flow through the system without being modified. Therefore, we do not have to worry about re-estimating or weighting the service time before we know their visit ratios. Each visit is assigned to exactly one class.

2. The routing rules are very clear to a modeler because it follows the natural job flow path. For instance, in the example given,

$q(o,1)=q(1,2)=q(2,3)=q(3,4)=q(4,5)=q(5,6)=q(6,7)=q(7,0)=1$

$q(k,j)=0$ otherwise.

Note that k in q(k,j) refers to <u>class number</u> instead of device number now.

This nice property enables us to solve for the visit ratios to each device <u>by inspection</u>. The transition matrix of such a system turns out to be almost diagonal. Section 3.4 gives the details.

Since we give each class a unique number, each class can be considered as an independent service station so that all the operating equations developed in the single-class model still holds in this multi-class model provided that the total utilizations of the classes of a device sums up to less than one.

3.2.3 Computational Algorithm for Closed System

Let c,i,j stand for classes and k for device. Define

U(k) : utilization rate of device k.

u(i) : utilization rate of class i to its corresponding device .

V(i) : visit ratio of class i to its corresponding device.

S(i) : service time of class i to its corresponding device.

Z(k) : the # of visits to device k.

D(k) : the average service time of device k.

C : the total # of classes in the network.

o : stands for the system or initial value.

b : stands for the bottleneck.

INPUT	STEP	Operating Equation	OUTPUT
q(i,j)'s	1.	$V(o)=1$ compute $V(j)=V(1)q(1,j)+V(2)q(2,j)+\ldots+V(C)q(C,j)$, $j=1,\ldots,C$	
S(i)'s	2.	compute $Z(k)D(k) = \sum_{i \in k} V(i)S(i)$ $k=1,\ldots,K$ $Z(b)D(b) = \text{Max}(Z(k)D(k))$ k in K. $X(o) = 1/Z(b)D(b)$	
	3.	compute $u(i),i=1,\ldots,C$ $U(k) = \sum_{i \in k} u(i)$, $k=1,\ldots,K$	U(k),k=1,...,K
N	4.	Compute R=N/X(o)	R

3.3 Multi-Transaction-Type, Multi-Class Queueing Network Model

3.3.1 Motivation

We have limited our discussion to the single transaction type queueing network systems which have either closed or open forms. In practice, there are normally many types of transactions that are being processed inside a computer system. For instance, consider the Baybank's X-press 24 hour service system which has to process cash withdraw, account balance, saving deposit, etc. Management may want to know the system's overall response time as well as each transaction's response time upon a customer's request. In a highly parallel computer systems, there are also many types of transactions being processed simultaneously. Therefore, it is desirable to have a general queueing network model to accommodate multiple transactions. One way to achieve this is to compute the weighted average service time and the weighted routing probabilities of all the transactions. Then treat the system as a single-transaction type system and apply the results that we developed previously for the single-class model to obtain the response time for each device and hence for each transaction type. This method, however, has two disadvantages :

1. The routing rules for different transaction types are different, especially in a highly parallel computer system. By merging different transaction types into a single transaction type, we also have to subjectively re- estimate the routing frequencies which will distort the real system we want to model. It may be easy to estimate the S(i)'s and q(i,j)'s for the first level devices, but certainly not clear to the lower level devices.

2. The model evaluator has to estimate the average of the average of transactions which reduces his confidence in the results. It is clear that the result obtained from this method will not be satisfactory.We will explore another method in the following subsection.

3.3.2 Modeling Technique

In a multi-class model, the multi-transaction type system can be modeled easily by simply considering the transaction types as different classes of customers that do no "talk" to one another. We briefly describe this idea for closed system here:

In a closed system, the number of customers in the system is fixed. There are several ways of modeling the problem using our single transaction type multi-class model. To list two of them :

1. Form a closed chain for each of the transactions.

2. Assign probabilities to each of the transaction types and form a single closed chain.

Note that in the first approach, the throughput of each transaction type may not follow some given quota but it will give us the best mix of transaction types. In the second approach, the transaction will follow a given quota which enables us to study the effect when given a mix of transaction types.

To study the interactions among transaction types, we can close other chains and single out a closed chain if the first approach is used or set probabilities to one for the interested transaction type if the second approach is used. Interactions among transactions can then be studied by checking whether they are additive.

We apply the second approach using the multi-class model to the P1L3 model of INFOPLEX with locality = .6 , read = 70 percents. The results are shown in the next section.

Note that the results are identical to the result from chapter 4 where RESQ is employed to compute the performance statistics for the same model. Hence, RESQ will serve as a validation for OPERA and on the other hand, OPERA can serve as a tool to communicate with and to convince management as well as an easy way to study the performance of INFOPLEX.

Fig3.4.1

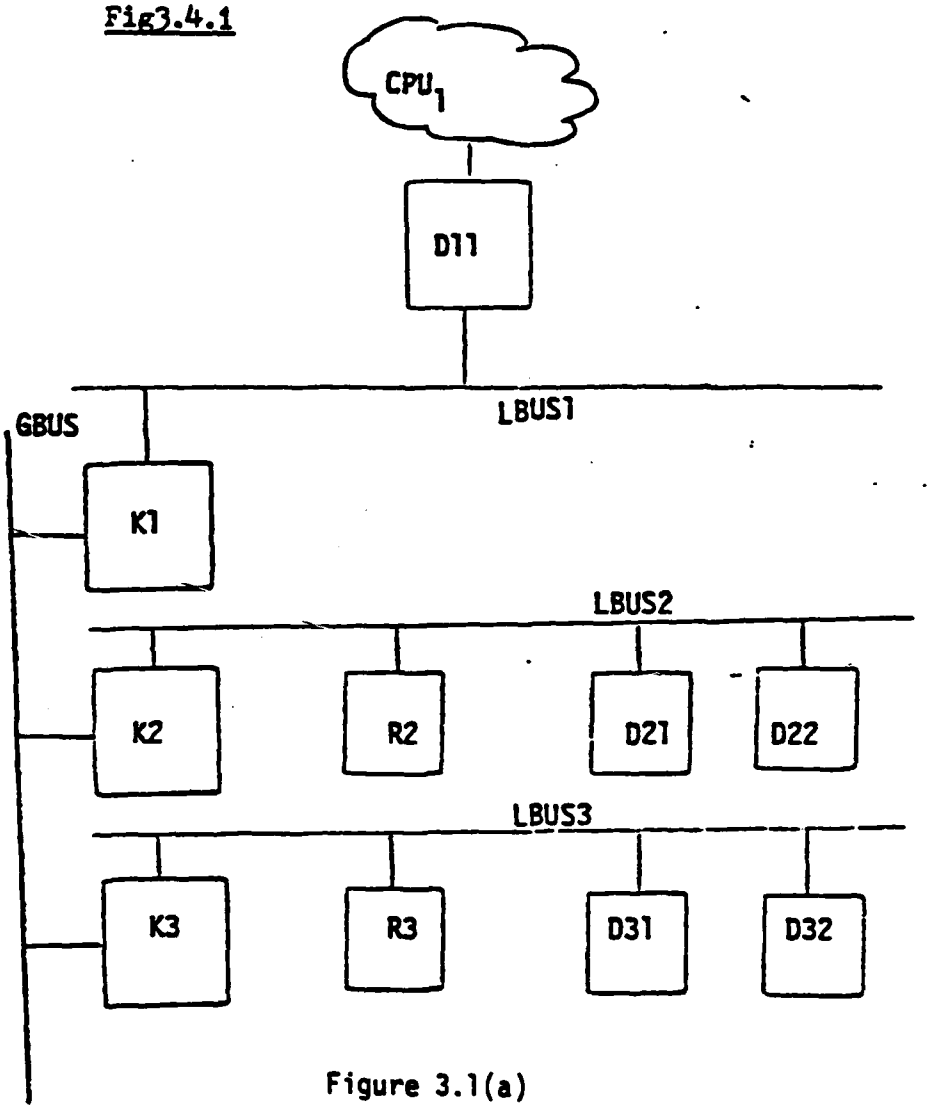


Figure 3.1(a)

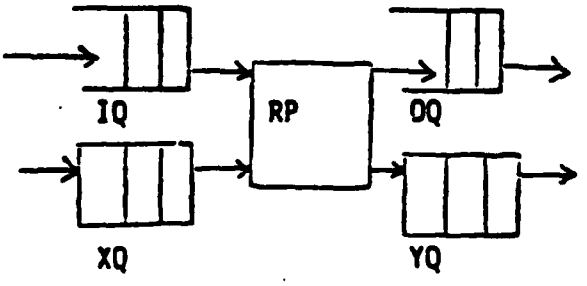


Figure 3.4.2

DEGREE OF MULTIPROGRAMING OF A CPU = 20

SIZES OF DATA QUEUES (XQ AND YQ) = 10

DIRECTORY SEARCH TIME = 200 NANOSEC.

READ/WRITE TIME OF A L(1) STORAGE DEVICE = 100 NANOSEC.

READ/WRITE TIME OF A L(2) DEVICE = 1000 NANOSEC.

READ/WRITE TIME OF A L(3) DEVICE = 10000 NANOSEC.

BUS SPEED = 10 MHZ

BUS WIDTH = 8 BYTES

SIZE OF A TRANSACTION WITHOUT DATA = 8 BYTES

BLOCK SIZE AT L(1) = 8 BYTES

BLOCK SIZE AT L(2 = 128 BYTES

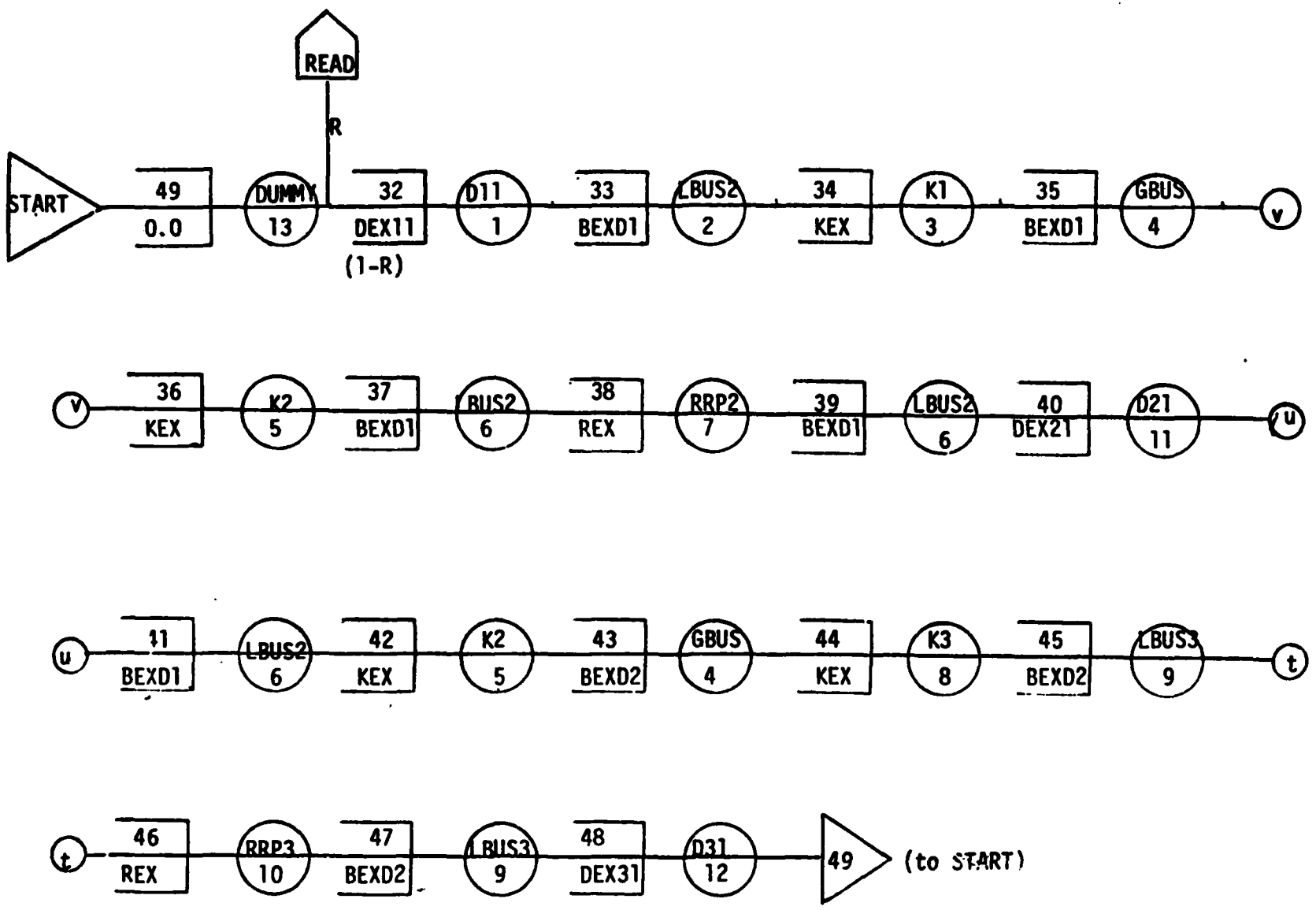
BLOCK SIZE AT L(3) = 1024 BYTES

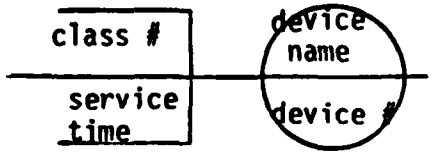
% READ REQUESTS = 70%

% WRITE REQUESTS = 30%

CONDITIONAL PROB. OF FINDING DATA IN A LEVEL

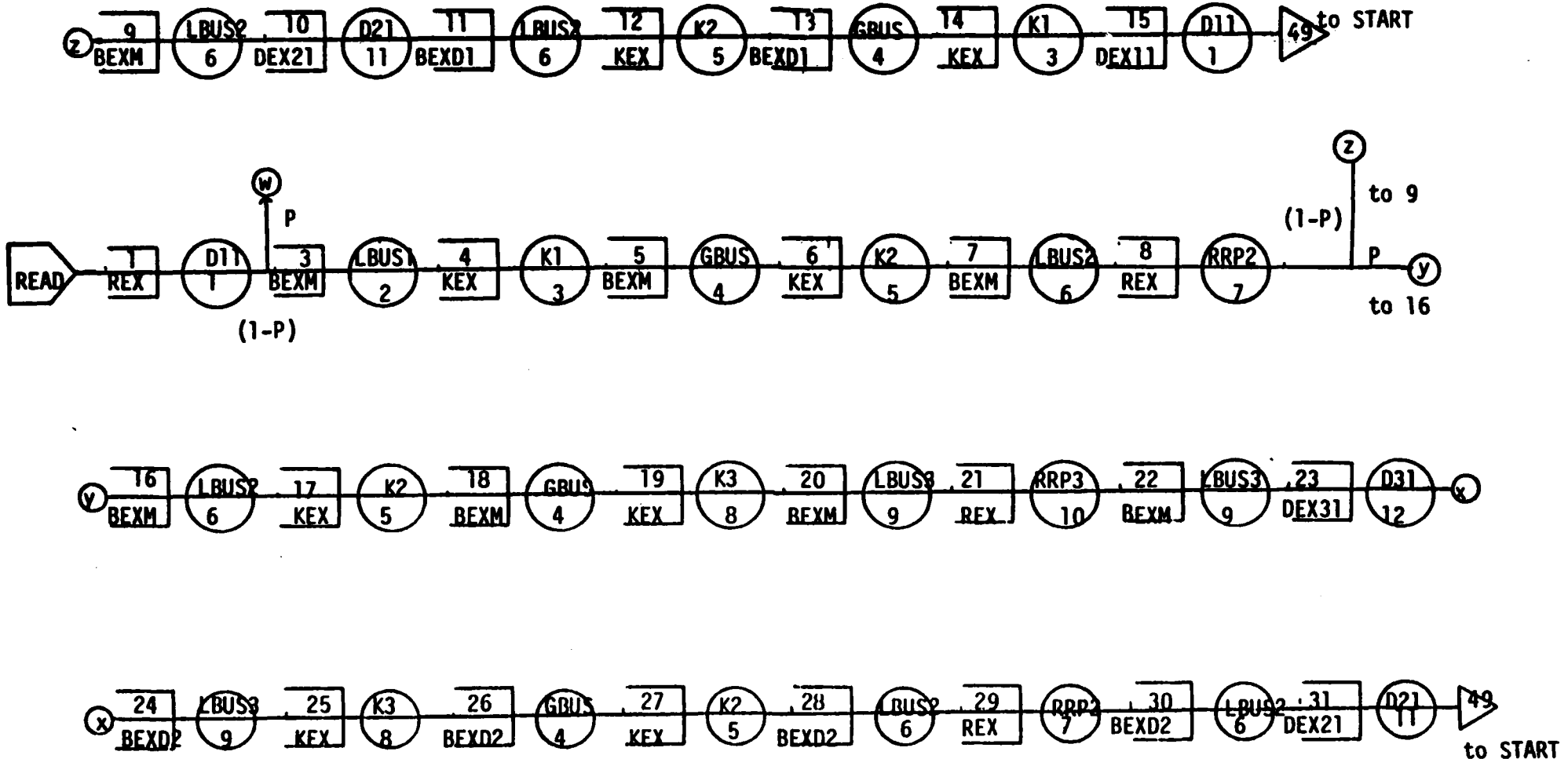
GIVEN THAT THE DATA IS NOT IN ANY UPPER LEVEL = P





WRITE PART; P1L3 Model

Figure 3.4.3(a)



READ P1L3 Model

Figure 3.4.3(b)

3.4 Computational Results of PlL3 Model of INFOPLEX Data Base Computer.

The PlL3 architecture is shown in Fig3.4.1 . The model is highly parametized. Parameters for the PlL3 model are chosen to reflect 1979 processor and storage technology. Two key parameters that characterize the references are the locality level and the proportion of read and write requests in the reference stream. The locality level (p) is the conditional probability that a reference is satisfied at a given storage level given that the reference is not satisfied in all upper storage levels. The concept used in 3.3 is applied to the PlL3 model using p=.6 and 70 percent read transactions. Fig3.4.2 summarizes all the model parameters.

The degree of multiprogramming is the maximum number of requests that can be active at a CPU. In the corresponding queueing network model, the degree of multiprogramming is used as the number of customers inside a closed system. The block size, bus speeds, bus width and speeds of the devices are parametized. We describe the mapping of the PlL3 model and workload features into a queueing network model in the next section.

3.4.1 Queueing Network Model Description

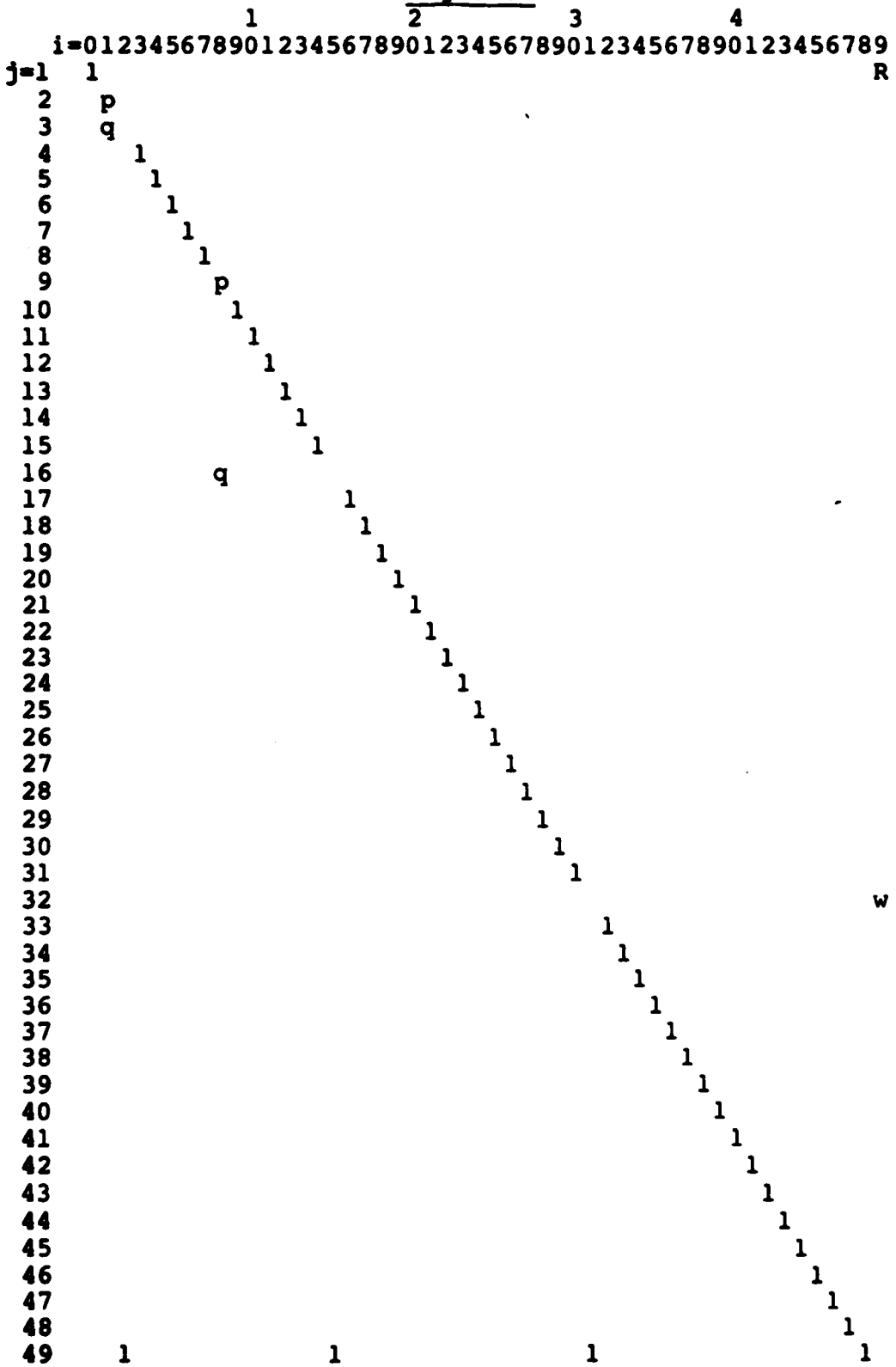
A mapping of the PlL3 model and its workload features into a queueing network model is shown in Fig3.4.3. Read operation and write operation are modelled separately by the single-transaction-type, multi-class modelling technique. The

logic of the model is discussed in Chapter 5, appendix, and briefly reviewed in 1.2. We highlight the modeling technique by discussing some of the operations.

A request to read a data item is handled by a data cache(D11,device 1) which has a directory service time REX(class 1 customer), it is retrieved (retrieve time for D11) at a read service time DEX11 (class2 customer) and sent to the processor(class49 which connects READ & WRITE operations together). This probability is characterized by Locality Reference(or p). If the data item is not in the data cache(i.e. not in device 1), the request is passed down to lower storage levels, one by one. Therefore, there is (1-p) probability that the read operation pass down to LBUS1(device 2) which has a message transfer time REXM(class3 customer). If the data item is found in the next level, it is returned through K1(device 3) back to D11(class15 customer with service time DEX11) and sent to the processor else the request is passed down to the next lower storage level. This is the basis for the mapping of the P1L3 READ operation and workload features into a queueing network model.

In a write operation, the data block to be updated is first read into the data cache(D11,class32 customer with service time DEX11). After the data block is updated, the data block is sent to the next lower storage level through LBUS1(device2),K1(device 3),GBUS(device 4),K2(device 5),LBUS2(device 6,class 37), RRP2(device 7), back to LBUS2(device 6,class 39), then to D21(device 11). Thus, the effect of the update is propagated to

Fig3.4.4



1 -> q(i,j)=1,R is portion of read(w=1-R),p=locality(q=1-p)

lower storage levels.

As mentioned in 3.3, the read submodel and the write submodel can be connected by a dummy device(device 13) with zero service time(see the front end of the write operation). Class 49 is used to connect the read and write operations together. R proportion of the transactions coming into the system are dispatched to the read submodel while (1-R) proportion of the transactions go to the wirte submodel.

This modeling technique ensures a clean decomposition among different transaction types.

Note that the class number can be used to help walking through the model. For instance, in the write model, class 32 customer is serviced at D11, then becomes class 33 customer, serviced by LBUS1 -> class 34 -> class 35 -> class 36 -> ... -> 48 -> 49. This is also the key to the diagonally structured transition matrix.

3.4.2 Computational Results

The computational results are shown below :

1. Compute V(c) :
   From Fig3.4.4 we see that given locality = .6, read =70%

   V(c=1)=.7, V(c=2)=.42
   V(c=3)=V(c=4)=...=V(c=8)=.28
   V(c=9)=V(c=10)=...=V(15)=.168
   V(c=16)=V(c=17)=...=V(c=31)=.112
   V(c=32)=V(c=33)=...=V(c=48)=.3

2. Compute X(o) : compute Z(k)D(k) for all devices
=> Z(k=12)D(k=12) is the bottleneck.
Z(b)D(b)=V(c=23)S(c=23)+V(c=48)S(c=48)

X(o)=1/{Z(b)D(b)} => X(o)=1/{.112*10000+.3*10000} = 1/4120
=243 transactions/milli-sec

therefore, throughput is 243 transactions per milli-second.

3. Compute U(k) for device k D(k) :

d(1)	class=	1	2	15	32		
S(c)		200	100	100	100		
V(c)		.7	.42	.168	.3		
U(c)		.0340	.0102	.0041	.0073		
U(k=1)			.056				
D(2)	class=	3	33				
S(c)		100	100				
V(c)		.28	.3				
U(c)		.0068	.00728				
U(k=2)		.014					
D(3)	class=	4	14	34			
S(c)		100	100	100			
V(c)		.28	.168	.3			
U(c)		.0068	.00408	.00728			
U(k=3)			.018				
D(4)	class=	5	13	18	26	35	43
S(c)		100	100	100	1600	100	1600
V(c)	.28	.168	.112	.112	.3	.3	
U(c)		.0068	.00408	.0027	.0435	.00728	.1165
U(k=4)				.181			
D(5)	class=	6	12	17	27	36	42
S(c)		100	100	100	100	100	100
V(c)		.28	.168	.112	.112	.3	.3
U(c)		.0068	.0041	.0027	.0027	.00728	.00728
U(k=5)				.031			

By the same token, we get

U(k=6)=.237
U(k=7)=.034
U(k=8)=.0127
U(k=9)=.282
U(k=10)=.020
U(k=11)=.141
U(k=12)=1.0 --- the bottleneck of the system.

4. Compute R=N/X(o) => R=20/{1/4120} = 82400
i.e. The system's response time is 82.4 micro-sec.

4 The RESQ Results Using PlL3 Model of INFOPLEX.

RESQ(research queueing analyzer) is a program package developed by M. Reiser and C.H. Sauer which provides the modeller with efficient methods of solution for a spectrum of queueing models of varying complexity, ranging from analytically tractable separable networks to rather general stochastic models. QNET4 is a subpackage of solution techniques of RESQ. It attempts to make available the most general class of queueing networks for which an efficient analytic solution can be obtained. This class is characterized by the existence of a product form solution. In order for a queueing network to have a product form solution, certain restrictions have to be imposed:

- stochastic, state independent routing .
- no explicit priorities.
- full accessibility(e.g. no blocking or finite queues).
- exponential service time distribution and poisson source.

General service time distributions are compatible with certain queue disciplines. In this paper, we use QNET4 solution technique to obtain results for the PlL3 model with the same model specifications. The queueing network model used is the same as Fig3.4.3 . We show the program for RESQ and the results computed from RESQ.

4.1 Model Description and Program.

The queueing network model that maps the PlL3 system and the corresponding workload features is exactly the same as 3.4.1 (see Fig3.4.3). This is intentionally done in order to make comparisons between different methods. The RESQ program is shown in Fig4.1.1 and

Fig4.1.1

```
TYPE P1L3RW RQ1DLOG
METHOD:UNCT
   OPEN CHAINS:0
   CLOSED CHAINS:1
   QUEUES:13
   CLASSES:49
COMMENTS?:YES
COMMENT:P1L3.WITH READ AND WRITE OPERATIONS
        :
CHAIN 1    TYPE:CLOSED
COMMENT:USE 49 TO CONNECT 1(READ CHAIN) AND
        :
(1):49->1;0.7000          32(WRITE CHAIN) TOGETHER.
(2):49->32;0.3000
(3):1->2;0.9000
(4):1->3;0.1000
(5):2->49
(6):3->4
(7):4->5
(8):5->6
(9):6->7
(10):7->8
(11):8->9;0.9000
(12):8->16;0.1000
(13):9->10
(14):10->11
(15):11->12
(16):12->13
(17):13->14          (35):31->49
(18):14->15          (36):32->33
(19):15->49          (37):33->34
(20):16->17          (38):34->35
(21):17->18          (39):35->36
(22):18->19          (40):36->37
(23):19->20          (41):37->38
(24):20->21          (42):38->39
(25):21->22          (43):39->40
(26):22->23          (44):40->41
(27):23->24          (45):41->42
(28):24->25          (46):42->43
(29):25->26          (47):43->44
(30):26->27          (48):44->45
(31):27->28          (49):45->46
(32):28->29          (50):46->47
(33):29->30          (51):47->48
(34):30->31          (52):48->49


CHANGE
MODEL NAME:
P1L3RW
FROM:
1
1->2 3; 7.00E-01 3.00E-01
:
1->2 3;.6 .4
FROM:
8
8->9 16; 7.00E-01 3.00E-01
:
8->9 16;.6 .4
FROM:
END OF CHANGES.
```

```
CHAIN POPULATION:20
QUEUE  1    TYPE:PS
COMMENT:D11 FOR CACHE
        :
RATE:1
CLASS LIST:1 2 15 32
WORK DMND. DISTR:200 100 100 100
QUEUE  2    TYPE:PS
COMMENT:LBUS1
        :
RATE:1
CLASS LIST:3 33
WORK DMND. DISTR:100 100
QUEUE  3    TYPE:PS
COMMENT:
RATE:1
CLASS LIST:4 14 34
WORK DMND. DISTR:100 100 100
QUEUE  4    TYPE:PS
COMMENT:
RATE:1
CLASS LIST:5 13 16 26 35 43
WORK DMND. DISTR:100 100 100 1600 100 1600
QUEUE  5    TYPE:PS
COMMENT:
RATE:1
CLASS LIST:6 12 17 27 36 42
WORK DMND. DISTR:100 100 100 100 100 100
QUEUE  6    TYPE:PS
COMMENT:
RATE:1
CLASS LIST:7 9 11 16 28 30 37 39 41
WORK DMND. DISTR:100 100 100 100 1600 1600 100 100 1600
QUEUE  7    TYPE:PS
COMMENT:
RATE:1
CLASS LIST:8 29 38
WORK DMND. DISTR:200 200 200
QUEUE  8    TYPE:PS
COMMENT:
RATE:1
CLASS LIST:19 25 44
WORK DMND. DISTR:100 100 100
QUEUE  9    TYPE:PS
COMMENT:
RATE:1
CLASS LIST:20 22 24 45 47
WORK DMND. DISTR:100 100 1600 1600 1600
QUEUE 10    TYPE:PS
COMMENT:
RATE:1
CLASS LIST:21 46
WORK DMND. DISTR:200 200
QUEUE 11    TYPE:PS
COMMENT:
RATE:1
CLASS LIST:10 31 40
WORK DMND. DISTR:1000 1000 1000
QUEUE 12    TYPE:PS
COMMENT:
RATE:1
CLASS LIST:23 48
WORK DMND. DISTR:10000 10000
QUEUE 13    TYPE:PS
COMMENT:
RATE:1
CLASS LIST:49
WORK DMND. DISTR:0
R; T=0.19/1.23 13:20:39
EVAL
MODEL NAME:
PIL3R1
VERSION DATE: MARCH 16, 1979
NO QNET ERRORS DETECTED.
```

Fig4.1.2

Fig 4.2.1

UT :

Q 1	N 1	N 2
0.056		
N 15	N 32	Q 2
		0.014
N 3	N 33	Q 3
		0.018
N 4	N 14	N 34
Q 4	N 5	N 13
0.181		
N 18	N 26	N 35
N 43	Q 5	N 6
	0.031	
N 12	N 17	N 27
N 36	N 42	Q 6
		0.236
N 7	N 9	N 11
N 16	N 28	N 30
N 37	N 39	N 41
Q 7	N 8	N 29
0.034		
N 38	Q 8	N 19
	0.013	
N 25	N 44	Q 9
		0.282
N 20	N 22	N 24
N 45	N 47	Q10
		0.020
N 21	N 46	Q11
		0.141
N 10	N 31	N 40
Q12	N 23	N 48
1.000		
Q13		
0.000		

TP :

Q 1	N 1	N 2
3.85E-04	1.70E-04	1.02E-04
N 15	N 32	Q 2
4.08E-05	7.28E-05	1.41E-04
N 3	N 33	Q 3
6.80E-05	7.28E-05	1.82E-04
N 4	N 14	N 34
6.80E-05	4.08E-05	7.28E-05
Q 4	N 5	N 13
3.09E-04	6.80E-05	4.08E-05
N 18	N 26	N 35
2.72E-05	2.72E-05	7.28E-05
N 43	Q 5	N 6
7.28E-05	3.09E-04	6.80E-05
N 12	N 17	N 27
4.08E-05	2.72E-05	2.72E-05
N 36	N 42	Q 6
7.28E-05	7.28E-05	4.50E-04
N 7	N 9	N 11
6.80E-05	4.08E-05	4.08E-05
N 16	N 28	N 30
2.72E-05	2.72E-05	2.72E-05
N 37	N 39	N 41
7.28E-05	7.28E-05	7.28E-05
Q 7	N 8	N 29
1.68E-04	6.80E-05	2.72E-05
N 38	Q 8	N 19
7.28E-05	1.27E-04	2.72E-05
N 25	N 44	Q 9
2.72E-05	7.28E-05	2.27E-04
N 20	N 22	N 24
2.72E-05	2.72E-05	2.72E-05
N 45	N 47	Q10
7.28E-05	7.28E-05	1.00E-04
N 21	N 46	Q11
2.72E-05	7.28E-05	1.41E-04
N 10	N 31	N 40
4.08E-05	2.72E-05	7.28E-05
Q12	N 23	N 48
1.00E-04	2.72E-05	7.28E-05
Q13		
2.43E-04		

Figure 4.2.2

RT:	CH 1N 1	CH 1N 2	CH 1N 3	CH 1N 4	CH 1N 5	CH 1N 6	CH 1N 7
	1.18E+05	1.96E+05	2.94E+05	2.94E+05	2.94E+05	2.94E+05	2.94E+05
	CH 1N 8	CH 1N 9	CH 1N 10	CH 1N 11	CH 1N 12	CH 1N 13	CH 1N 14
	2.94E+05	4.90E+05	4.89E+05	4.90E+05	4.90E+05	4.90E+05	4.90E+05
	CH 1N 15	CH 1N 16	CH 1N 17	CH 1N 18	CH 1N 19	CH 1N 20	CH 1N 21
	4.90E+05	7.36E+05	7.36E+05	7.36E+05	7.36E+05	7.36E+05	7.36E+05
	CH 1N 22	CH 1N 23	CH 1N 24	CH 1N 25	CH 1N 26	CH 1N 27	CH 1N 28
	7.36E+05	5.48E+05	7.33E+05	7.36E+05	7.34E+05	7.36E+05	7.34E+05
	CH 1N 29	CH 1N 30	CH 1N 31	CH 1N 32	CH 1N 33	CH 1N 34	CH 1N 35
	7.36E+05	7.34E+05	7.35E+05	2.75E+05	2.75E+05	2.75E+05	2.75E+05
	CH 1N 36	CH 1N 37	CH 1N 38	CH 1N 39	CH 1N 40	CH 1N 41	CH 1N 42
	2.75E+05	2.75E+05	2.74E+05	2.75E+05	2.74E+05	2.73E+05	2.75E+05
	CH 1N 43	CH 1N 44	CH 1N 45	CH 1N 46	CH 1N 47	CH 1N 48	CH 1N 49
	2.73E+05	2.75E+05	2.72E+05	2.74E+05	2.72E+05	8.74E+04	8.24E+04

Fig4.1.2. The program is self explanatory: It has a closed chain, 13 queues, 49 classes of customers. The model has read and write operations with class 49 connecting class 1 (read chain) and class 32(write chain) together. Class 49 has R=.7 chance to go to the read chain and (1-R)=.3 chance to go to the write chain. Class 1 has locality p=.9 to go to class 2 and p=1-.9=.1 chance to go to class 3. Note that this can be easily changed to .7 and then .6 as shown in the CHANGE part in Fig4.1.1 which makes sensitivity analysis easy once the model is set up.

Workload features for each device is set up in Fig4.1.2. Consider Queue 1 (D11 for cache): It is a processor sharing device which has 4 classes of customers, namely class1,2,15,and 32. The service time for these devices are 200,100,100, and 100 respectively. Rate: 1 is redundent in this case.

By the same token, the workload features can be set up for all devices. This can be done easily when referring to the queueing network model in Fig3.4.3.

4.2 Comparision of RESQ Results to the Corresponding OPERA Results.

The results of P1L3 for R=.7,p=.6 are shown in Fig4.2.1 and Fig4.2.2. Since the utilization rate is a measure of how much a device is utilized, it should be on the

device level instead of the class level. Comparing the utilization rates in Fig4.2.1 vs the results in chapter 3.4.2, we see

that they are identical to the third digit. They are listed here again :

	Q1	Q2	Q3	Q4	Q5	Q6
OPERA	.056	.014	.018	.181	.031	.237
RESQ	.056	.014	.018	.181	.031	.236

	Q7	Q8	Q9	Q10	Q11	Q12
OPERA	.034	.013	.282	.020	.141	1.0
RESQ	.034	.013	.282	.020	.141	1.0

Throughputs of each device are computed and printed out under the TP section. In interpreting this output, it is clear that the throughput at Q13(the dummy device with zero service time) should be used to represent the system's total throughput. The throughput can be looked up from the table which gives :

2.43E-04 transactions/nanosec. => 243 transactions/millisec. (exactly the same as what we ahd from OPERA).

Response times of all devices are also available under the RT section. Since class 49 is the entry to the system, the response time can be read off under CH1,N49 which is 8.24E+04, again exactly the same as the OPERA result from 3.4.2.

We have shown that RESQ's result confirms with OPERA's result. Since the RESQ code is unavailable, it is uncertain whether the formulae used by RESQ are identical to OPERA's or OPERA turns out to be a special case to the RESQ approach. This is not a critical issue though. The important part of the ball game is that now we have an intuitive (easy to explain to the managements), hand calculable(and

even by inspection only, see chapter 6) method to estimate the performance statistics of the complex INFOPLEX data base computer.

5 Lam's Results Using P1L3 Model of INFOPLEX

Lam79 evaluated the performance statistics of P1L3 model. A listing of the P1L3 GPSS simulation model is presented in appendix. To illustratee the model logic, the following is a brief deescription of the path followed by a read-through transaction. A read request (TXN) is queued in KIQ3 (the input message queue of the storage level controller at level 3). When KRP3 is free, TXN is serviced and put in KOQ3, When LBUS3 is available, TXN is sent to RIQ3 (the input message queue of the memory request processor at level 3) where it waits for RRP3, the request processor. RRP3 then searches its directory to obtain the real address for TXN. TXN is put into ROQ3 to be sent to a storage device,say D31. When LBUS3 is free, TXN is sent to DIQ31. TXN waits in DIQ31 (the input message queue for device D31) for DRP31 to be free and also for a slot in DYQ31 (the output data queue for D31) to hold the retrieved data. When both conditions are met, DRP31 retrieves the data and puts it in DYQ31 where it waits for the LBUS3 to be free and for there to be a slot inKXQ3 (the input data queue of the storage level controller at level 3) to hold the data. When both conditions are met, the data is sent to KXQ3. Then the data is put in KYQ3 waiting for the GBUS and for all the upper storage levels to be free to receive the broadcast. When these conditions are met, the data is broadcasted to all upper storage levels. At the highest storage level, the data is sent to the appropriate data cache controller which forwards the data to the CPU. (Lam79)

Figure 5.1.1

Nbar	Locality Level (P)	Throughput (per msec)	Mean Response Time (NSec)	Utilizations						
				GBUS	LBUS1	DATA CACHE	LBUS2	D21	LBUS3	D31
18.3	.20	286	64032	.42	.07	.10	.63	.11	.52	1.00
18.2	.30	320	56908	.42	.07	.12	.60	.12	.52	1.00
17.8	.40	456	39142	.45	.08	.16	.63	.15	.59	1.00
17.2	.50	548	31324	.50	.10	.20	.65	.17	.62	1.00
18.9	.60	698	27114	.51	.10	.23	.63	.19	.65	1.00
17.1	.65	758	22505	.51	.10	.26	.62	.18	.68	1.00
18.9	.70	811	23317	.53	.10	.27	.65	.20	.69	1.00
15.4	.80	947	16298	.50	.94	.31	.57	.19	.71	.99
3.6	.85	598	6021	.23	.52	.19	.26	.09	.35	.52
2.3	.90	581	3957	.16	.04	.17	.19	.06	.26	.42
2.12	.95	532	3986	.14	.03	.15	.16	.05	.21	.25

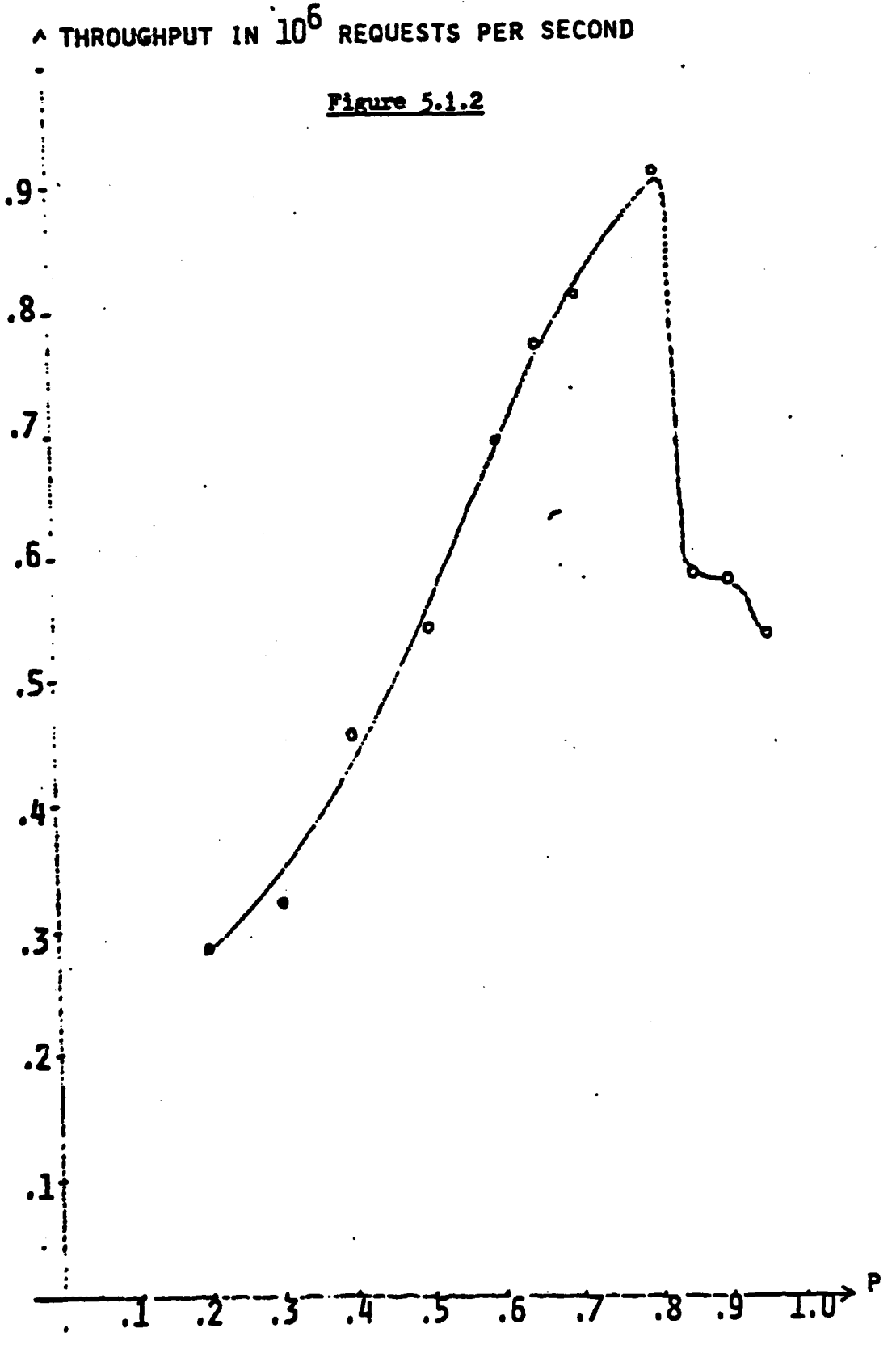
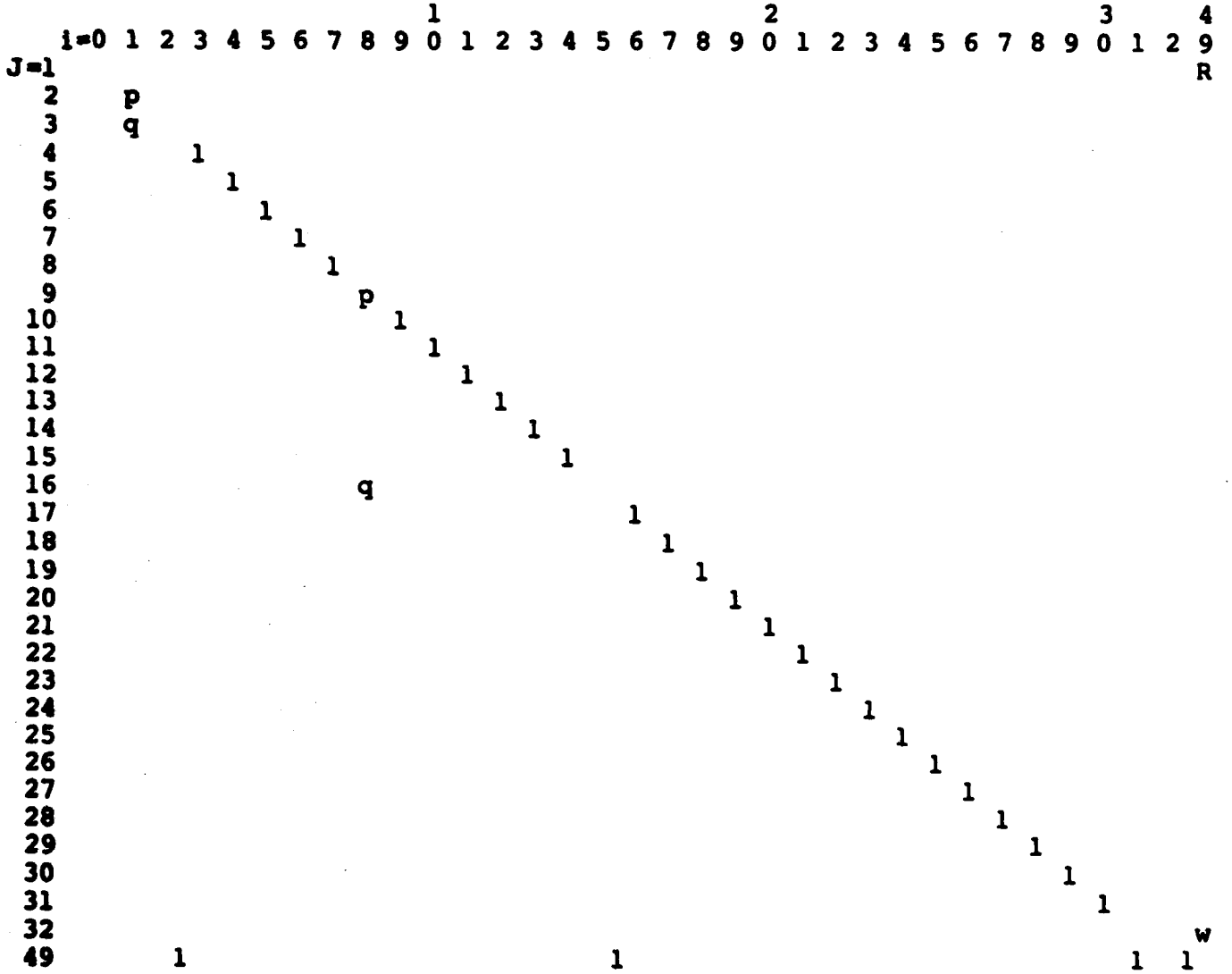

^ THROUGHPUT IN 10^6 REQUESTS PER SECOND
.9
.8
.7
.6
.5
.4
.3
.2
.1
.1
.2
.3
.4
.5
.6
.7
.8
.9
1.0
P

Figure 5.1.2

5.1 Lam's Results of P1L3 of INFOPLEX.

A series of simulations was carried out to obtain data points by varying the locality levels. The results of these simulations are presented in Fig5.1.1. Throughputs are plotted against locality levels in Fig5.1.2. In general, as the locality level increases, throughput also increases. A throughput of close to one million transactions is obtainable at about p=.80 locality level. However, after the p=.80 point, throughput drops sharply as the locality level increases. This is caused by the deadlock phenomenon.

Fig5.2.1



Note : 1 means q(i,j)=1 ; R is the proportion of read transaction ; w = 1 - R

p = locality level ; q = 1 - p ; p,q,w,R means q(i,j)=p,q,w,R.

B=.7

Z \ P	.2	.3	.4	.5	.6	.65	.7	.8	.85	.9	.95	
Z1	195.2	206	215	223	229	231	238	237	238	238	240	D11
Z2	56	49	42	35	28	25	21	14	11	7	3.5	LBUS1
Z3	67.2	64	59	53	44	40	36	25	19	13	7	K1
Z4	828.8	647	487	350	235	186	143	73	46	25	10	GBUS
Z5	156.8	132	109	86	67	58	48	31	23	15	7	K2
Z6	1556.8	1210	907	648	431	275	258	129	80	43	16	LBUS2
Z7	201.6	167	134	105	78	66	55	34	24	15	7	RRP2
Z8	89.6	69	50	35	22	17	13	6	3	1	.35	K3
Z9	717.7	617	454	315	211	154	113	50	28	13	3.15	LBUS3
Z10	89.6	69	50	35	22.4	17.2	13	5.6	3	1.4	.35	RRP3
Z11	560	492	420	350	280	244	210	140	105	70	35	D21
Z12	4480	3430	2520	1750	1120	858	630	280	1575	70	18	D31
Z(b)	4480	3430	2520	1750	1120	858	630	280	238	238	240	bottle neck
X(o)	223.2	292	397	571	893	1166	1587	3571	4202	4202	4167	/milli
R	89605	68610	50400	35000	22400	17150	12600	5600	4760	4760	4800	nano-sec
Lam's												
X(o)	286	320	456	548	698	758	811	947	598	581	532	/milli
R	64032	56908	39142	31324	27114	22505	23317	16298	6021	3957	3986	nano-sec

Figure 5.2.2

5.2 Comparision of Lam's Results with OPERA Results.

Since the definitions of transaction completion time are different between Lam's and the queueing network model developed in Fig3.4.3, it is necessary to modify the Fig3.4.3 model in order to be comparable to Lam's model.

By simplifying the write submodel to a trivial model, namely :

49 -> (with chance 1-R) 32 -> 49, we can easily obtain the comparable queueing network model. The transition matrix is shown in Figure 5.2.1. Computations were done for different locality levels with 70 percent read transactions. The results are shown in Fig5.2.2.

The throughputs and response times are listed in the lower part of Fig5.2.2 for both Lam and OPERA. The throughput are again plotted in Fig6.1.1. It is exciting to see that the the results are comparable to within a factor of 2 except the cases where simulation ran into the deadlock situation. This clearly indicates that by employing the intuitive and cost effective OPERA technique, the researcher can obtain a rough estimate for the INFOPLEX model under investigation.

In the next chapter, we extend the OPERA model to a general DSH-11 model and show that by inspection, we can obtain the ceiling throughput for an INFOPLEX model provided that the locality level is not high enough to switch the bottleneck to another device.

6 Extension of OPERA to General DSH-11 Models.

We have so far discussed OPERA with multi-transaction-type, multi-class modeling technique and have computed performance statistics for PlL3 model for different situations. Simulation and RESQ's results provide us with confidence in our analysis.

An interesting question to be answered is that : Given a PqL1 model with R proportion of read transactions and locality level p, what is the maximum throughput for the model ?

6.1 A General Formula of System Throughput for DSH-11.

From the experiences obtained previously, it is clear that if the service time of device at L(i+1) is bigger than L(i) by a significant factor,say 10, then when the system reaches steady state, the lowest level device will always be the bottleneck provided that the locality reference p is not high enough to switch the bottleneck to another device.

From the diagonally structured matrix, we see that for the queueing network model used in chapter 4, the corresponding OPERA's system throughput is computed by the formula below :

$$X(o) = 1/\{Z(b)D(b)\} = 1/\{V(\text{read part})S(b)+V(\text{write part})S(b)\}$$

$$= 1/(\{R(1-p)^{**}(3-1) + w\}*S(b))$$

where S(b) is the service time of the bottleneck device.

This relation holds for 1 level case. Therefore, the system's throughput for PqL1 model with (R,p,S(b)) is given by :

$$X(o) = 1/(\{R(1-p)^{**}(1-1)+w\}*S(b)) \text{ where } w=1-R$$

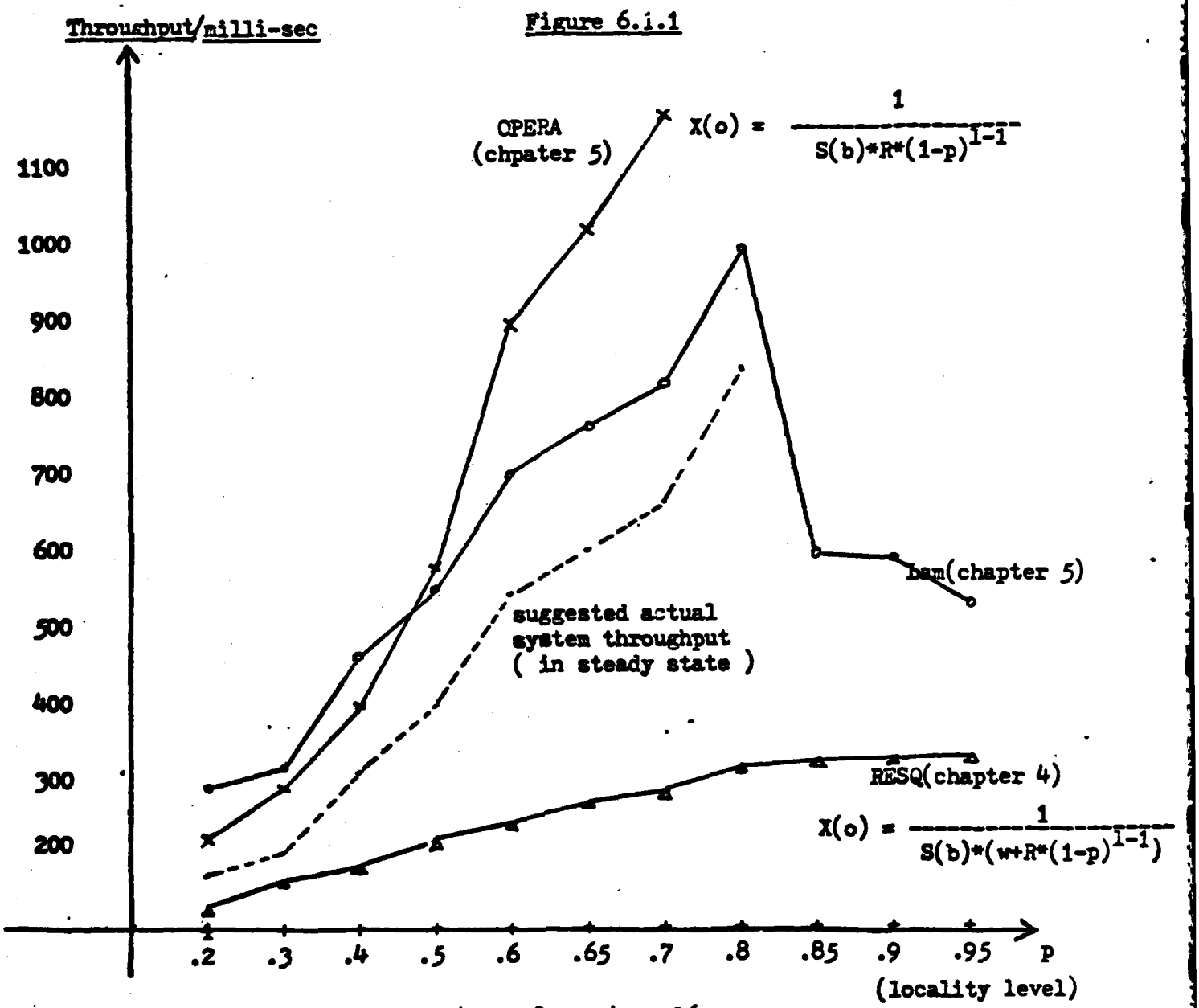
In the chapter 5 model where write operation is considered to be complete once the first level is updated because of the support of the store-behind algorithm, we have w=0.

In the chapter 4 case, as p increases, the system's throughput asymptotically approaches 1/{w*S(b)}. In the chapter 5 model where w=0, the system's throughput is is dominated by the L(1) device which is the slowest one. However, as p increases, (1-p)**(1-1) becomes very small which means only a very small portion of the transactions will be propagated to the lowest level. Most of the transactions will be handled on the first level because of the high locality. It is possible tha t the bottleneck will switch to D11 or other devices according to the specific design and workload features. From the DSH-11 design point, we like to be careful about this.

We restate our observation here : For a PqL1 model of DSH-11, when the closed system reaches steady state, the system's throughput is bounded by the bottleneck device. If :

1. There are sufficient customers inside the system to cause a bottleneck situation.

Figure 6.i.1



Lam-OPERA : .22 .09 .13 -.04 -.28 -.54 -.96

legend: R = 70% (proportion of READ transactions)
X(o): System throughput.
S(b): slowest device service time.
w = 1-R
Data for PiL3 model

2. The lowest level device is also the slowest device by a significant factor.

3. The locality level p is not high enough to switch the bottleneck.

4. The traffic of any bus is not heavy enough to cause that bus to be bottleneck.

then the system's throughput can be estimated by :

$$X(o) = 1/\{(R\{(1-p)^{**}(1-1)\} + w)*S(b)\}$$

For the Chapter 5 model, we have w=0.

Fig6.1.1 plots the throughputs for the chapter 4 model and the chapter 5 model as a function of locality p. It is clear that :

1. The chapter 4 model asumptotically approaches 1/{w*S(b)}.

2. The chapter 5 model is an exponential function of (1-p)**(1-1) at low locality and becomes concave at high locality (p > .8) because the switch of the bottleneck (from D31 to D11).

6.2 Ceiling Throughput of a DSH-11 Model.

It is worth noticing that the queueing network model we constructed has the nice property of possessing a ceiling throughput(an upper bound) to the system under question.This property is ascribed to the following relaxations that we set for the model :

1. Overflow handling is ignored. We assume that no existing block has to be evicted in order to accomodate an incoming block.

2. Broadcasting operation is ignored.(this condition can be included by generating some comparable transactions to certain devices.)

3. Infinite queue storage size is assumed.

4. No priority is assumed.

5. Store-behind operation is ignored in the chapter 5 model. Write operation is processed in "no time".

With these constraints relaxed, it is clear that the throughput obtained from this kind of model should serve as a ceiling throughput to any refined model. For instance, the throughput from a simulation model should be smaller than the throughput from the simplified queueing network model because a well developed simulation model would take more constraints into consideration, hence possesses a tighter bound to the system.

6.3 Actual System Throughput of a DSH-11 Model.

From the arguments in 6.2, it is very clear that the throughputs obatined from OPERA should serve as upper bounds to Lam's throughputs using simulation models. Two more evidences support this view:

1. Initially a system is empty, transactions will be processed at a faster speed, response time will be faster and throughput higher. As time goes by, response time should slow down, throughput decrease . Their product should be equal to Nbar, the average number of customers in the system. If a system was not simulated long enough to get rid of the initial conditions, the simulated throughput will be higher than the actual throughput and on the other hand simulated response time be faster. This explains why some of the simulated throughputs are higher than the OPERA's results.

2. Bottleneck analysis emphasizes the behavior of the bottleneck device. When a bottleneck situation occurs, the service times at non-bottlenecks are assumed to be zeros to obtain the

bottleneck throughput. Hence the throughput obtained from this analysis should be higher than the actual throughput. Response time, on the other hand, is faster than the actual response time because all non-bottleneck are like "super-conductor" --- not delaying any work at bottleneck at all.

Fig6.1.1 depicts the proposed actual system throughputs. If one computes the Nbar's for Lam's simulation, he will definitely find that the Nbar's are always smaller than the actual number of customers in the system. This is proved to be true (see Fig5.1.1). Furthermore, it should be true that the closer Nbar is to the actual number, the better the simulation results will be. We will investigate further and support this view with evidences(to appear in technical report) The arguments made in this chapter are observational and intuitive. It is interesting that by using the OPERA coupled with the multi-class, multi-transaction-type modeling technique, we not only avoid the complex mathematics in the stochastic queueing network analysis, but also "see through" the complex DSH-11 model by identifying the bottleneck. Conclusion of this report is presented in the next chapter where discussion of future directions along this line of work is also made.

7 Conclusion and Future Directions.

This paper presented an easy to understand, cost effective, and analytically tractable solution which yields sufficient accuracy for performance questions. The success of this solution method is ascribed to the operational analysis framework and the multi-transaction, multi-class concept. The result computed from this method is close to the results from both the simulation approach used by Lam and the RESQ approach which uses stochastic distribution and complex mathematics.

It is clear that the simple algorithm we used in this paper can be used as a primary tool for the evaluation of an interested system. If the result turns out to be interesting, we then construct programs for either RESQ or GPSS, etc., to further confirm and extract the performance statistics for INFOPLEX.

A lot of work remains to be done in this area. For instance, does the diagonally structured transition matrix hold in general? If so, why, if not, can we modify our technique so that the the visit ratios can always be read off by inspection ? Secondly, what if the number of customers inside the DSH-11 system is not large enough to cause a bottleneck situation? Thirdly, wouldn't it be more natural to consider an open system rather than a closed system? If an open system is used, how are we going to validate our algorithms and results? Should we simulate the same model or build a real system? Fourthly, can a concrete and formal approach be employed to solidify the intuitions and arguments that we made on chapter 6? And although we have obtained a ceiling throughput, is this upper bound close to an optimal solution ?

How can we obtain a tighter bound based on the work that we have done so far? Finally, is there a standard approach to decomposing and mapping a system into a queueing network model ?

Performance evaluation is an indispensable ring in the design of the INFOPLEX data base computer. Simulation is always a useful technique to employ but it is uncertain when the system will reach steady state. The huge amount of CPU time required , in addition to the model building time, to obtain steady state performance statistics makes it less attractive to the researcher. Traditional queueing netwrok analysis involves a lot of stochastic assumptions and complex mathematics which are difficult to understand and to gain insights. Oftentimes researchers devote a lot but gain very little out of it. The operational analysis offers a good opportunity for modeler to make intuitive arguments and to use the results with more confidence and understanding. It is possible to devise a method of solution and some modeling technique which provides the modeler with enough flexibility and confidence to map a system and its workload features into a queueing network model and then solve for an answer yielding sufficient accuracy for many performance questions. Future work in this area should be aimed toward this goal.

8 References INFOPLEX Part:

(Lam79) Lam C.Y. & Madnick,S.E. : Technical Report 4, Simulation Studies of the DSH-11 Data Storage Hierarchy System. September 1979.

(Lam and Madnick,1979a): Lam,C.Y., and Madnick,S.E., "Technical Report No. 1: The Intelligent Memory System Architecture - Research Directions",M.I.T. Sloan School Internal Report No.M010-7908-01,August 1979.

(Lam and Madnick,1979b) : Lam,C.Y., and Madnick , S.E., "Technical Report No. 2 : The IMS Data Storage Hierarchy - DSH-1", M.I.T. Sloan School Internal Report No. M010-7908-02,August 1979.

(Lam and Madnick,1979c): Lam,C.Y., and Madnick,S.E.,"Technical Report No.3: The IMS Data Storage Hierarchy - DSH-11",M.I.T. Sloan School Internal Report No. M010-7908-03,August 1979.

(Madnick,1975a) : Madnick,S.E., "Design of a General Hierarchical Storage System", IEEE INTERCON Proceedings,1975, 1-7.

(Madnick,1977) : Madnick,S.E.,"Trends in Computers and Computing : The Information Utility",Science,195,4283(1973),1191-1199.

(madnick,1979): Madnick,S.E., "The INFOPLEX Database Computer: Concepts and Directions",Proceedings IEEE Computer Conference, February 26,1979,168-176.

(Madnick80): Private discussion between Madnick and the author.

Analytic Model Part:

(Bard80) Bard,Y "Some extensions to multiclass queueing network analysis" Proc. 4th Int. Symp. Model. & Perf. Eval. Comp. Syst. IFIPWG7.3 Noth-Holland,Netherlands February 1979

(Bask75) Baskett, F.;Chandy, K.M.;Muntz,R.R.; & Palacios,J. " Open, closed, and mixed networks with different classes of customers," J. ACM 22,2,April 1975,248-260.

(Buzen75) : Buzen,J.P. "Cost effective analytic tools for computer performance evaluation," in Proc. IEEE COMPCON,1975. IEEE,New York,pp293-296.

(Buzen76a) : Buzen,J.P. "Fundamental opeational laws of computer system performance," Acta Inf. 7,2(1976),167-182.

(Buzen76b) Buzen,J.P. "Operational analysis :the key to the new generation of performance prediction tools," in Proc.IEEE COMPCON,1976, IEEE New York.

(Buzen78a) Buzen,J.P. "OPerational analysis: an alternative to stochastic modeling," in Proc.Int. Conf. Performance Computer Installations,1978,North-holland Pub. Co., Amsterdam, The

Netherlands,pp.175-194.

(Buzen78b) Buzen,J,P.,et.al. "BEST/1-design of a tool for computer system capacity planning," in Proc. 1978 AFIPS National Computer Conf.,Vol.47,AFIPS Press, Montvale,N.J.,pp.447-455.

(Buzen78c) Denning,P.J. & Buzen,J.P. : "The Operational analysis of queueing network models", Computing Surveys,Vol.10,No.3,Sept 1978.

(Buzen80) Buzzen,J.P. and Denning P.J. : " Operational treatment of queue distributionb and mean-value analysis" Computer performance Vol 1 no 1 June 1980.

Chandy,K.M., and R.T. Yeh : "Current trends in programming methodology, vol. III software modeling Printice Hall 1978.

(Infotexh) Infotech state of the art report on performance modeling and prediction, Inoftech Int. Ltd., Maidenhead, UK,1977

(Klein75) Kleinrock,L. Queueing Systems I, JOhn Wiley,New York,1975.

(Klein76) Kleinrock,L. Queueing Systems II, John Wiley, New York,1976.

(Lavenberg80) Lavenberg,S S and Reiser M : "Stationary state probabilities at arrival instants for closed queueing networks with multiple types of customers" IBM T.J. Watson Research Center,NY,USA, Report RC 7592,April 1979.

(Reis75) Reiser, M.: "Queueing networks with multiple closed chains : theory and computation algorithms," IBM J. Res. Dev. 19; May 1975,283-294.

(Reiser78) Reiser,M and Sauer,C.H., "Queueing network models:methods of solution and their program implementations", in Current trends in programming methodology III,K.M. Chandy and R. Yeh, Printice Hall, Englewood Cliffs,N.J.,1978,p115-167.

(Reiser80) Reiser,M and Lavenberg,S S " Mena value analysis of closed multichain queueing networks",J.Assoc. Comput. Mach. Vol 27,No 2; April 1980.

(Scher67) Scherr,A.L. An analysis of time shared computer systems, MIT press, Cambridge, Mass., 1967.